A floating polygon soup representation for 3D video

Thomas Colleu

N° d'ordre : 4142 **ANNÉE 2010**

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Traitement du signal

Ecole doctorale MATISSE

présentée par

Thomas Colleu

préparée à l'unité de recherche IRISA / INRIA Rennes Bretagne Atlantique en collaboration avec l'unité de recherche IETR et à Orange Labs - France Télécom R&D, Rennes

---

A floating polygon soup representation for 3D video

Thèse soutenue à l'IRISA, Rennes le lundi 6 Décembre 2010

devant le jury composé de :

Béatrice PESQUET-POPESCU
Professeur, Télécom ParisTech / *rapporteur*

Marc POLLEFEYS
Professeur, ETH Zurich / *rapporteur*

Mohamed DAOUDI
Professeur, Télécom Lille 1 / *président*

Vincent CHARVILLAT
Professeur, ENSEEIHT-IRIT / *examinateur*

Claude LABIT
Directeur de recherche, INRIA / *examinateur*

Luce MORIN
Professeur, INSA-IETR / *directeur de thèse*

Stéphane PATEUX
Ingénieur R&D, Orange Labs / *co-directeur de thèse*

Acknowledgments

At the heart of this adventure, I would like to thank the duo Luce Morin and Stéphane Pateux for having guided my research with so much expertise and enthusiasm. Luce and Stéphane, while our regular meetings always provided me valuable feedbacks and inspiration, you also supported me when I needed and carefully reviewed all writings and slides. I owe you my sincere gratitude for all these reasons.

I also want to thank Professor Marc Pollefeys and Professor Béatrice Pesquet-Popescu for kindly accepting to review my dissertation and for being part of the jury. Many thanks also to Professor Vincent Charvillat and Professor Mohamed Daoudi for being part of the jury and providing comments and suggestions. I also express my thanks to Claude Labit for being part of the jury and for his feedbacks throughout these three years.

I would like to thank Christine Guillemot leader of the TEMICS team and Ludovic Noblet leader of the CVA team for giving me the opportunity to work towards a Ph.D degree. I also want to thank all members and ex-members of these teams, particularly Raphaele Balter who supervised my work with Stéphane and Luce and whose Ph.D thesis was a source of inspiration; Gael Sourimant for showing me the way; my office mates Vincent Jantet, Josselin Gautier and Jérôme Allasia all three for fruitful discussions as well as sometimes pointless relaxing ones; Dorra Riahi for her contribution within INSA/IETR.

Many thanks to all colleagues and friends who all contributed at some point to this thesis (in pseudo-random order): Mathieu Urvoy, Mathieu Moinard, Mathieu Desoubeaux, Mathieu Rubeaux, Angélique Drémeau, Shasha Shi, Cédric Herzet, Mehmet Turkan, Yohan Pitrey, Simon Bos, Julien Fayolle, Ana Charpentier, Jonathan Taquet, Joachim Zepeda, Jean-Marc Thiesse, Simon Malinowski, Velotiaray Toto-Zarasoa, Fuchun Xie, Laurent Guillo, Huguette Béchu, Olivier Lemeur, Jean-Jacques Fuchs, Caroline Fontaine, Teddy Furon, Aline Roumy, Patrick Heas, Maxime Pelcat, Isabelle Amonou, Nathalie Cammas, Maryline Clare, Gordon Clare, Christophe Daguet, Patrick Gioia, Joel Houssais, Joel Jung, Gilles Teniou, Philippe Vonwyl, (please forgive me if I forgot you).

I cannot thank my wife enough, Dina, for her precious support and releasing words during this thesis and particularly during hard times.

I also express great thanks and gratefulness to my parents, brother and sister as well as all my friends for their never ending encouragements.

Contents

Chapter 1

Introduction

Contents

1.1 Context

The year 2010 has seen the popularity of 3D video exploding. Starting with the success of 3D movie 'Avatar' in cinemas in December 2009[1] and with electronic companies announcements of their 3D-ready televisions arriving at home[2]. Indeed technologies have been sufficiently improved so that two views of the same scene can be captured at the same time, processed, transmitted and displayed with good quality. Here, the functionality that justifies the term '3D' is stereoscopy [Whe38]. It exploits the binocular vision of humans such that a better perception of depth is obtained, giving a sensation of relief[3]. In fact, there is only one more image compared with traditional 2D video. But it is sufficient to show the potential of 3D video to improve the description of a scene and the feeling of depth for the users.

Active research is now focused on **multi-view video** in order to increase the number of images of the same scene at the same time [SKS05, Mag05, KSM+07, DKM+10]. Multi-view video brings mainly two functionalities to the users. The first functionality is free viewpoint navigation: similarly to the famous "bullet time effect" in the movie 'The matrix', the viewer can change the point-of-view within a restricted area in the scene, thus having a navigation functionality. The second functionality is auto-stereoscopy: the viewer enjoys stereoscopic visualization without the use of special glasses which is a great advantage over actual stereoscopic solutions. Although auto-stereoscopy already works with two views, it is restricted to only one viewer who must stand still at the

[1] `http://avatarblog.typepad.com/avatar-blog/2010/01/avatar-biggest-movie-of-alltime.html`

[2] `http://www.3dtvsource.com/sony-2010-3d-tv-lineup/`

[3] (french) `http://www.3d-tvee.com/technologies-3d/explication-de-la-stereoscopie-en-video-351`

correct position in front of the screen. On the contrary multi-view auto-stereoscopy brings much more freedom in terms of viewing position and number of users[4].

With free viewpoint navigation and/or auto-stereoscopy, several **applications** are possible. An application providing navigation functionalities is free viewpoint television (FTV): a remote, a mouse or any other user interface is used to control the viewpoint. If a user tracking device is used, then the viewpoint can change automatically according to the user's position. As the user moves, he can perceive the depth of the scene, as demonstrated by Lee in his demo video[5]. 3DTV is another application that takes advantage of auto-stereoscopic visualization: a multi-view video is watched in 3D without special glasses, providing that a multi-view auto-stereoscopic display is used (figure 1.1). These applications are very promising and would not only be intended for entertainment but also for medical domain, video conferencing or training as few examples.

Figure 1.1: Example of the relief sensation in a 3DTV application using a multi-view auto-stereoscopic TV by Philips.

Setting up a **multi-view video system** raises many challenges at every stage of the system. Figure 1.2 represents the different stages of a multi-view video system. First, multiple views are captured during the acquisition stage, then the data is processed and formated during the representation stage. These two stages are performed at the operator side. During the transmission stage, compression/decompression of the data is usually required in order to minimize the data load and thus reduce the transmission bandwidth and storage space. At the user side, view synthesis may be performed to generate all the views required by the display device. Finally, the views are displayed to the user in a certain way depending on the display device and the user preferences.

The **goal of this study** is to focus on the representation stage. The representation

[4] `http://www.alioscopy.com/3d-solutions-displays`
[5] `http://www.youtube.com/watch?v=Jd3-eiid-Uw`

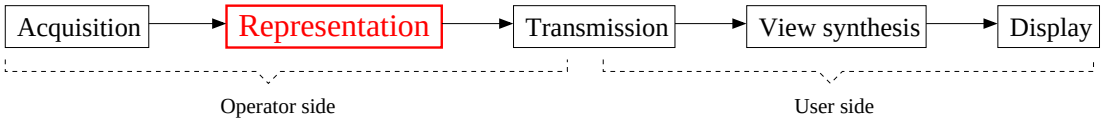


Figure 1.2: Overview of a multi-view video system. The representation stage is the focus of this study.

plays a central role in a multi-view video system. Indeed, it influences the data load to be transmitted, the compression method to be used, as well as the computational complexity during the view synthesis stage and the final video quality at the display stage. Existing representations for multi-view video often contain some geometric information about the scene in addition to color information. They all exhibit advantages but also drawbacks at some point in the system. We want to analyze these pros and cons, and propose a new representation that takes into account, in a unified manner, the different issues such as data load, compression compatibility, computational complexity and image quality.

1.2 Thesis outline

This thesis is organized into eight chapters. *Chapter 3* presents a study of existing representations in the context of multi-view video. The conclusions that come out from this study list a few features that matter for the choice of a representation. *Chapter 4* introduces our proposed representation and its properties. The remaining chapters are dedicated to the construction and validation of this representation. *Chapter 5* explains the construction of the representation. *Chapter 6* presents the view synthesis stage and evaluates the quality of synthesized images. *Chapter 7* introduces a new compression method adapted to the proposed representation. Results of this compression method are compared with an existing approach studied in the MPEG's 3DV group. Finally, *chapter 8* introduces a new method for reducing texture misalignments in synthesized views when using a geometric model. This method is applied and evaluated on the polygon soup.

Chapter 2

Challenges in multi-view video systems

Contents

To better understand the challenges raised by the development of a multi-view video system, more details about each stage shown in figure 1.2 are now given. Some important properties appear several times throughout the system such as the computational complexity, the data load, the navigation range and the image quality. Usually, technical choices have to be made to manage all the compromises induced by these properties. The last section of this chapter introduces some constraints on the multi-view video system according to the targeted application scenarios, namely transmission of multi-view video for FTV or 3DTV.

2.1 Acquisition

An acquisition system for multi-view video is usually made of multiple cameras and possibly other devices like depth cameras, spotlights, microphones... Many technical choices have to be made such as the characteristics of the cameras (resolution, frame rate); how they are arranged together (small baseline, sparse setup); whether they should be mobile or static; and also how they are synchronized and networked. All these choices are generally guided by non-technical constraints: the application scenario may require large free viewpoint navigation or high quality auto-stereoscopic visualization;

the scene content may be only indoor in a controlled environment, or outdoor; also financial and logistical constraints play an important role in the choice of the acquisition system.

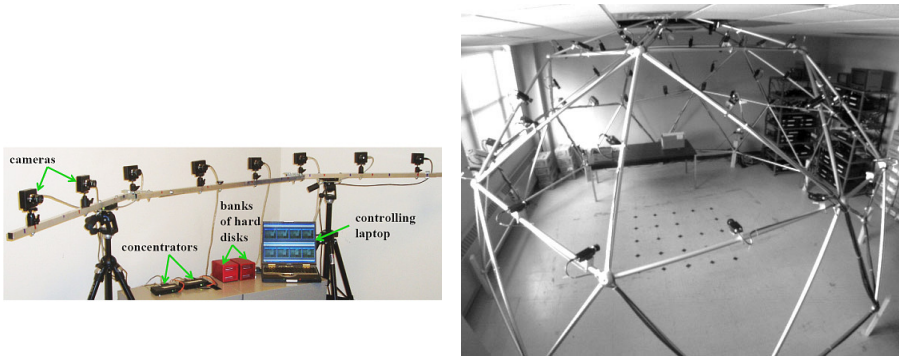
The **camera arrangement** is particularly important. For auto-stereoscopic visualization, cameras should be placed very close to each other at a distance equivalent to the eye's distance (about 6.5 cm). This is a small baseline and specific stereoscopic cameras have been designed for this purpose (figure 2.1). With this type of camera

(a) An 8-cameras system by 3DTV solutions (b) A 15-cameras system by Heinrich Herz Institut (HHI) [FMZ+08]

Figure 2.1: Examples of small baseline camera systems.

arrangement, cameras are often in a parallel setup such that they lie side-by-side on a straight rig, and their image planes are coplanar. This results in horizontal-only parallax from one image to another. This set-up has proved to provide higher stereoscopic image quality [WDK93, YOO06]. Resulting images are said to be rectified, however since physical rectification is not perfect, an additional image rectification algorithm is often applied to the images during or after the acquisition process [PKVG99, FTV00]. Real-time processing of image rectification can be achieved if implemented on graphics hardware [YPYW04]. In addition, cameras may not have the same color when capturing the same object, therefore color calibration [JWV+05] and color correction [SJYH10, TISK10] are also important for better color consistency between the cameras .

Increasing the number of cameras also increases the complexity of the system in terms of computation complexity as well as data load for transmission. Therefore, large free viewpoint navigation is difficult to obtain with such small baseline systems. An alternative is to increase the distance between cameras so that to find a compromise between the system complexity, the desired navigation range, and the final video quality (figure 2.2). In this case, intermediate views between cameras have to be computed using so-called view synthesis algorithms as explained in section 2.4. To do so, cameras must be related to each other, i.e. exact position and orientation of the cameras must be estimated. This process is called **geometric camera calibration** [Tsa86, Zha00].



(a) An 8-cameras system by microsoft [ZKU+04]

(b) A 51-cameras system surrounding the scene by Carnegie Mellon University [RNK97]

Figure 2.2: Examples of medium or wide camera arrangement systems.

Another type of camera is gaining interest for multi-view video systems. It is called **depth camera** (a.k.a z-camera or time-of-flight camera[1]). An example of such camera is shown in figure 2.3. This type of camera captures an image of the distance of the scene: laser beams (often in infrared spectrum) are emitted to the scene, and the reflections are collected by the device to measure the time of flight. Another alternative to laser beams is structured light that illuminates the scene with alternating pattern that helps recovering the depth of the scene [WWCG07]. With these kinds of Z-cameras the resulting so-called depth map provides important information about the geometry of the scene. This geometry is particularly useful for intermediate view synthesis. When no depth camera is available, geometry is usually computed with complex multi-view reconstruction algorithms. Since depth cameras provide real-time depth maps, it can be used as a complement or replacement of multi-view reconstruction algorithms [KBKL09].

Figure 2.3: A depth camera by PMDTechnologies.

[1] `http://en.wikipedia.org/wiki/Time-of-flight_camera`

2.2 Representation

During the representation stage, the captured multi-view video is processed and formated to obtain a certain representation of the data. This representation often contains some geometric information about the scene in addition to color information. The choice of a representation plays a central role in a multi-view video system. It influences the data load to be transmitted, the compression method to be used, as well as the computational complexity during the view synthesis stage and the final video quality at the display stage. This subject is the heart of this thesis report since a new representation will be introduced. A more detailed study of existing representations will be given in the next chapter.

2.3 Transmission

When transmitting the data over a network, it should be transfered fast enough to provide comfortable visualization at the user side, without big delays. However, compared to traditional single-view video, the amount of data with multiple views is increased and the networks may be overloaded. For video, the amount of data is defined by the number of bits per second (bps) and it is called the bit rate. Therefore, in order to reduce the bit rate, **compression methods** adapted to the representation are particularly important and challenging [Say05, SKS05, MMW07].

It is also of great importance to define compression **standards** ensuring that videos can be watched by as many users as possible, just like language is a standard for communication. A well known compression standard is the JPEG one, used for compressing images. Concerning video compression, two working groups of experts are developing standards: MPEG (Moving Picture Expert Group) and VCEG (Video Coding Expert Group). These two groups collaborated in the Joint Video Team (JVT) for defining video compression standards (e.g. AVC, SVC, MVC). Within MPEG organization, a sub-group called 3DV is currently studying the representation and compression of multi-view data for 3D video applications. In addition, the MPEG's group 3DGC (3D graphics compression) is working on 3D graphics and mesh compression standards. Existing standards are: MPEG-4 AVC for single-view video compression; MPEG-4 MVC for multi-view video compression; MPEG-4 AFX for 3D graphics tools; and MPEG-4 3DMC for 3D mesh compression.

One of the main principle of compression methods is to **remove correlation** in the data and encode the representation. For video compression, the spatial and temporal correlation between pixels is exploited [Gal92, WSBL03]. For example, between two successive images of a video sequence, two similar blocks of pixels can be related by a motion vector. Thus, a block of pixels can be predicted by a block in the previous frame and a motion vector. Similarly, for multi-view video compression, inter-view correlation (i.e. correlation between the multiple views) can also be removed in addition to spatial and temporal one [MSMW07a]. Finally the same principles can also be adapted to the compression of 3D geometric data [DBD08].

Many compression methods, like JPEG and MPEG, are lossy, i.e. some information

is lost in order to reach a target bit rate. This loss involves **distortions** in the data. These distortions should be small enough to prevent annoying artifacts. Therefore, the performance of lossy compression methods are often evaluated with the compromise between bit rate and distortions (figure 2.4). Moreover, since artifacts vary depending on the properties of the data, special attention is required to adapt the compression method to the data properties. This is particularly true for color images and depth maps: both are pixel images but with different characteristics that require adapted compression methods. A lot of research activities have been recently dedicated to compression methods adapted to the properties of the depth maps [MM09, MMS$^+$09, Mor09, YV09].

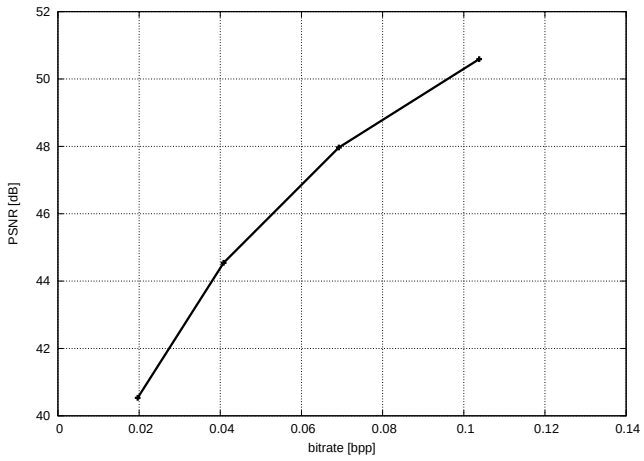


Figure 2.4: A rate-distortion curve used to evaluate the performances of a lossy compression method. The x-axis gives the bit rate in bits per pixel (bpp) for an image or bits per second (bps) for video sequence. The y-axis gives the image image quality generally expressed in decibels (dB) using the so-called PSNR measure. Other quality measures could be used. A good compression method should exhibit as high quality as possible, i.e. low distortion, for a given bit rate or as low bit rate as possible for a given quality.

2.4 View-synthesis

As mentioned above in the acquisition stage, **intermediate views** between cameras are frequently needed for free viewpoint navigation or auto-stereoscopic visualization (figure 2.5). These intermediate views (or virtual views) are computed with image-based rendering (IBR) algorithms. They use the original images and possibly additional geometry information and camera parameters [KLTS06, Mag05]. If only original images are used, then the parallax effect cannot be reproduced into intermediate views. Parallax (or disparity) is "the depth-dependent projection of the 3D scene into the image when the view point changes" [SKS05]. On the other hand, if geometry and camera

parameters are available then intermediate views can be synthesized between existing cameras with wide camera arrangement.

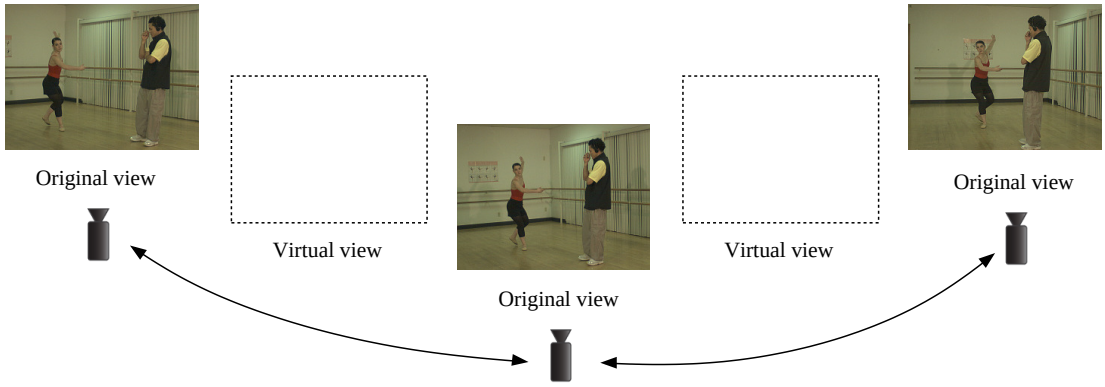


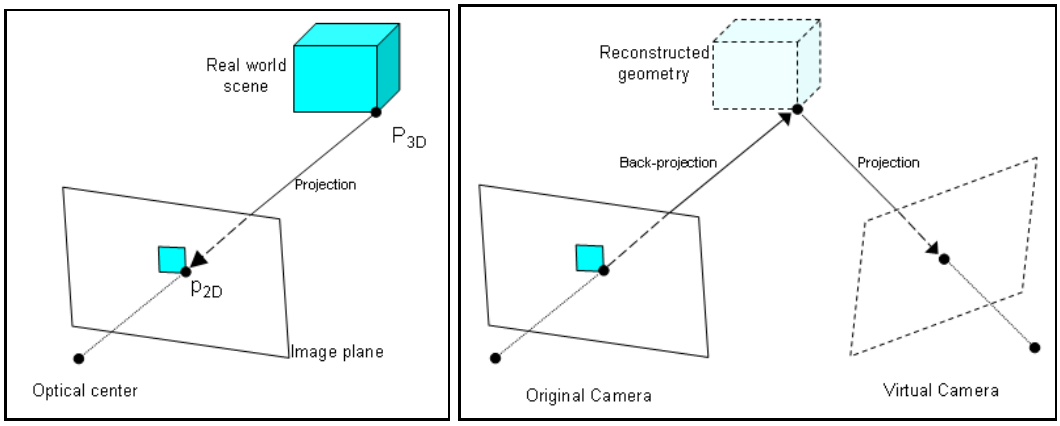
Figure 2.5: Virtual views are often needed for free viewpoint or auto-stereoscopic visualization.

Modeling geometry and camera parameters are two challenging tasks. The first is often referred as multi-view reconstruction[2] [SCD+06] or stereo analysis[3] [SSZ01, Sou10] and the second as camera calibration [Tsa86, Zha00]. Indeed, although huge progress has been made in this field, real world scenes and acquisition devices are still difficult to be accurately modeled, and some typical situations still lead to reconstruction errors and camera calibration errors. Moreover, such algorithms usually require a lot of computations which is always a drawback for fast processing of the data. This explains the growing interest for depth cameras that could be used as a complement or replacement to multi-view reconstruction algorithms.

Virtual view synthesis is mainly based on **projection principles**. Figure 2.6(a) shows the projection of a 3D object into the image plane of a camera. The creation of a pixel in the image plane of a camera can be mathematically modeled using the position of the 3D point in the scene as well as the camera position and internal parameters. This process is called projection. Therefore, a virtual view can be synthesized as shown in figure 2.6(b): a 3D point on the reconstructed geometry is projected into the image plane of a virtual camera. The color of the 3D point is first computed by inverse projection (back-projection) of the original image into the 3D space. If not points but polygons are used as 3D primitives, then the color of the image is mapped onto the polygon, which is called texture mapping.

During this projection process, special care has to be taken about **occlusions**, i.e. when a foreground object occludes the background when viewed from a given viewpoint. Indeed, in this case occluded 3D point may be projected into the same pixel position as non-occluded one, but only the front 3D point with smallest depth must be displayed in the image. Using a Z-buffer or an occlusion compatible scanning order are two solutions for this issue [Mor09].

[2] `http://vision.middlebury.edu/mview/`
[3] `http://vision.middlebury.edu/stereo/`



(a) Image capture (b) Virtual view synthesis

Figure 2.6: Projection principles. (a) Image capture. Projection of a real-world 3D point into the image plane of the camera resulting in a 2D pixel. (b) Virtual view synthesis. Back-projection of a pixel to the reconstructed geometry and projection into a virtual camera.

For synthesizing good quality virtual views, not one but multiple original views are usually projected, which raises the question of the **blending** of these multiple views projected into the virtual view. Some original views may be less reliable than others depending on the camera position, field of view and resolution of the camera. Therefore, lower weights should be associated to these views during the blending process. Such strategy has been studied by Debevec et al. [DTM96] called 'view-dependent texture mapping' and by Buehler et al. [BBM+01] called 'unstructured lumigraph rendering'. Moreover, when multiple geometries are used, possible inconsistencies between these geometries create disturbing artifacts, particularly around depth discontinuities. These artifacts are known as **ghosting artifacts** or corona artifacts and require special processing to be removed.

Finally, **additional processing** of the virtual view are often needed for hiding remaining artifacts. Especially, some parts of the 3D scene may be visible in the virtual view but not in the original views because of occlusions by foreground objects. When synthesizing a virtual view, these parts become disoccluded, resulting in pixels with unknown color. These pixels should be filled using some filtering [SMD+08] or inpainting [OYH09, Dar09] methods that exploit neighboring pixels in order to predict the unknown ones.

All the above computations needed for intermediate view synthesis represent a certain **complexity**. Since real-time visualization is usually desired for interactive 3D video experience, the view synthesis complexity should always be compatible with the computation power of the device. Thanks to recent graphics hardware and programming, most view-synthesis methods can run in real-time. This is because of the high parallel processing capabilities of such hardware. However, if no powerful graphics hardware or any other specialized parallel devices are available, e.g. on a mobile device,

then only low complexity view-synthesis methods can be performed.

Once one or multiple virtual views have been synthesized, a typical problem is the **quality assessment** of these images [WB06]. Subjective evaluation is generally considered as the most reliable method: a group of observers give their opinion about the quality of the images. The result is given by the mean opinion score (MOS). However, this method is expensive and slow, thus it cannot be systematically used. Instead, objective methods for automatic evaluation exist. The aim is to apply a computational model that can measure the image quality accurately and automatically. Several objective measures have been developed depending on whether full reference, reduced reference or no reference images are available for comparison. However, designing such measures that can predict the way that human visual systems perceive images is still a challenging task, particularly if the images are used for stereoscopic visualization since two images are viewed at the same time. In order to evaluate the quality of virtual views, the most popular method is to compare each view with an original view as reference: a virtual view is synthesized at the same position and orientation as an original one such that a full-reference quality measure can be used. An often-used quality measure is the peak signal-to-noise ratio (PSNR), although this measure does not take into account the human visual system neither the structure nor statistics of natural images.

2.5 Display

The last stage of a multi-view video system is the display of the images to the users [KH07]. If only free viewpoint navigation is wanted, then the display may be any classical 2D screen and navigation would be provided by some user interface like a remote, a mouse or even a tracking device, along with adapted processing hardware. However, if stereoscopic visualization is desired, special displays are needed. The challenges here are to provide the user with an accurate description of the scene with a natural sensation of depth. Moreover, comfort of visualization and freedom of movements are important to prevent fatigue. Finally, the cost of the device is an important criterion for acceptation into the mass market.

Today's most popular displays are based on the **stereoscopy principle**: two images of the scene are displayed, they correspond to the two views of the eyes (figure 2.7). However, to ensure that each eye sees only one image, special glasses must be worn. Two different techniques are mainly used: alternate frame sequencing of the left and right images with synchronized shutter glasses and a high frame rate display, or projection with polarizing glasses and polarizing projectors. The main drawback with these solutions is that glasses are a source of discomfort and reduce the luminosity of the images.

In order to eliminate the use of special glasses, **auto-stereoscopic screens** have been developed [Dod05]. Instead of glasses, a system that separates images is placed directly in front of the screen. Two concurrent systems are currently used: parallax barrier or lenticular lenses (figure 2.8). With this kind of technology, the viewer should be at the exact position where each eye sees one image, which would be a strong

(a) A 3D TV (200 Hz refresh rate) and shutter glasses by sony.

(b) A 3D polarizing projector with polarizing glasses by LG.

Figure 2.7: Examples of stereoscopic displays.

constraint in practice. Therefore two solutions exist: those that use head-tracking to adapt the images to the viewer position, and those that display multiple views creating multiple viewing zone so that no tracking is needed. This second solution is called multi-view auto-stereoscopic display and present the advantage that multiple users can view the images at the same time. However, a drawback is that the resolution of the images is reduced proportionally to the number of views displayed at the same time. Another drawback, shared with glasses-wearing displays, is that stereoscopic visualization provides only horizontal parallax. It means that the eyes must be aligned horizontally, thus the user cannot tilt his head or lie on a couch. Prototypes of full parallax displays are being developed and show promising results [JK10]. Up to now, multi-view auto-stereoscopic displays are mainly intended for professional customers, but manufacturers plan introducing them to the mass market in the future. The order of navigation for today's multi-view auto-stereoscopic displays range about 5 to 30.

Auto-stereoscopic screens do not produce real 3D images but provide a sensation of depth. On the other hand, **volumetric displays** [Fav05] and **holographic displays** [SCS05] provide a more detailed description of the scene by either filling the 3D space with imagery or reproducing the light wavefront reflected by the scene. So far, such systems are under development and not yet mature enough.

2.6 Introduction of constraints on the targeted system

We have seen in this chapter that many choices have to be made at all the stages of the processing chain. This section introduces some constraints on the multi-view video system that will be considered in the following. These constraints are guided by the targeted application scenarios (transmission of multi-view video for Free viewpoint TV or 3DTV), and by some choices concerning the trade-offs that appear throughout the system. These constraints will help comparing existing representations with each others

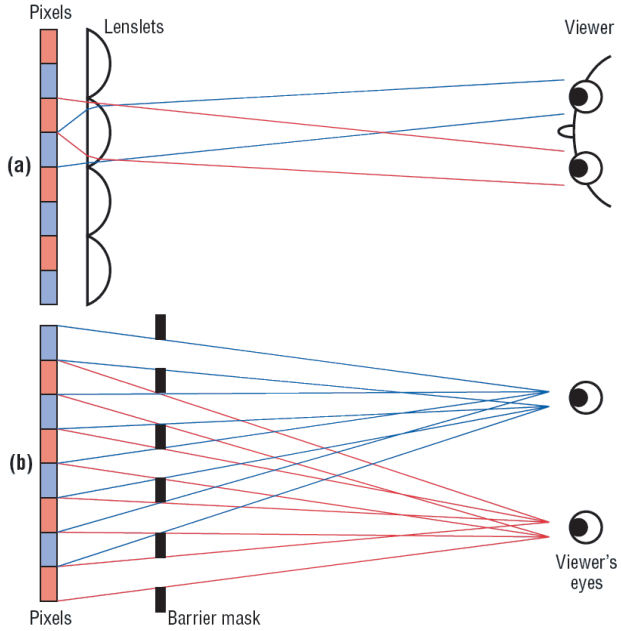


Figure 2.8: Principle of two-views autostereoscopic displays (Image extracted from [Dod05]). (a) Lenticular: the lenslets direct the pixels' light so that each of the viewer's eyes sees light from only every second pixel. (b) Parallax barrier: a barrier mask is placed in front of the pixel raster so that each eye sees light from every second pixel.

according to this defined system and then help to propose a new representation.

- Acquisition: At the acquisition stage, many choices have to be made to cope with the trade-offs between the number of views, the range of navigation and the computational complexity of the system. Since free viewpoint navigation is desired, and transmission is done over a band limited channel, then a medium or wide camera arrangement setup should be used and thus intermediate views need to be synthesized. This helps limiting the number of cameras while providing a given navigation range, but this increases the computational complexity of the system. In addition, the system should be adapted to any arbitrary scene.

- Representation: The need for view synthesis increases the construction complexity of the representation because some geometry of the scene has to be estimated. In this study, no constraint about the construction complexity is defined, on the other hand low complexity at the user side is necessary for real-time applications.

- Transmission: The data would be transmitted over a band limited network.

- View synthesis: At the user side, important constraints are real-time and high quality visualization. Although actual auto-stereoscopic displays require rectified images such that only horizontal parallax is possible, we do not restrict the navigation in order to provide more freedom of navigation and compatibility with future full parallax displays. However, this arbitrary configuration induces more complexity at the view synthesis stage. Moreover, for real-time view synthesis, we do not put constraints upon having 3D graphic devices (e.g. GPU) at the user side.

- Display: Concerning the display device, visualization over multiple types of displays should be possible. 2D display could be used in addition with free viewpoint functionalities. Multi-view auto-stereoscopic displays with horizontal or even full parallax could be used. A combination of multi-view auto-stereoscopic displays and free viewpoint functionalities is also possible.

Acquisition	**Representation**	**Transmission**	**View-synthesis**	**Display**
Medium to wide camera arrangement	No complexity constraint about the construction of the representation	Transmission over band limited channel	Real-time view synthesis in all directions (full parallax)	Multiple types of displays
Arbitrary scene			GPU available if need	Navigation functionality

Table 2.1: Summary of constraints on the multi-view video system considered in this study.

Chapter 3

Existing representations

Contents

This chapter contains a study of existing representations for multi-view video. The goal is to analyze the pros and cons of each representation. First, the definition of a representation is given and the general properties of a representation are introduced. Then some existing representations are detailed together with associated end-to-end systems. Here, this is a general study such that the constraints on the system defined previously are not taken into account. The representations are classified into four families of representations: image-based, depth image-based, surface based and impostor based representations. Finally, the pros and cons of each representation are analyzed according to the constrained system that we have defined. Additional material about 3D video and its data representations can be found in [SKS05, Mag05, SMM+09, AYG+07, Smo10].

3.1 Definition of a representation

In the context of multi-view video, a representation is the **description of the scene** using a certain type of data. For example, an image describes the texture of the scene from a certain viewpoint, therefore an image is a representation. Similarly, a depth map is the representation of the geometry of a scene. Texture and geometry are the main information used to represent a scene. They can be dynamic to incorporate the motion of the scene. Additional information such as illumination models and camera parameters may be added to the representation.

The choice of the representation is of central importance in a multi-view video system. First the representation is computed from the acquisition data, and then it is compressed and transmitted. At the receiver side, the representation is decompressed, and rendered on a display device. At each processing stage, the **properties** of the representation may be adapted to a certain application and usually raise some trade-off. It is important to analyze these properties before concluding if it is adapted to a desired application. Figure 3.1, gives general representation properties corresponding to each processing stage of a 3D video system. In the following, these properties are discussed.

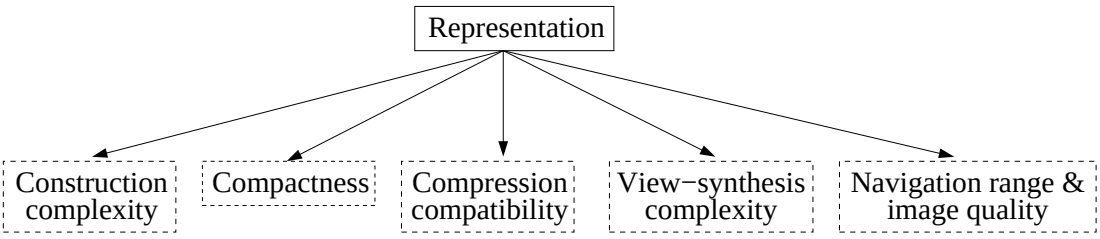


Figure 3.1: Representation properties

Construction complexity. The construction complexity corresponds to the amount of processing needed to construct the representation. Estimating the geometry of the scene for example requires many computations. It influences the processing time of the system. If real-time processing is required, such as in 3D video conferencing or live 3DTV broadcasting, then a representation with low construction complexity should be used. However this requires to do some compromise at the user side of the system. Indeed, such low complexity representation may not contain enough information to provide navigation functionality or all the complexity may be transferred at the user side which may reduce interactivity or real-time visualization.

Compactness. The compactness corresponds to the amount of physical data stored in the representation. On the contrary to the compression process where the data is seen as a stream of bits, here the data has a physical meaning in the scene such as color or depth value. Representations using multiple textures and/or multiple geometries contain redundancies and are less compact than representations using a single texture

and/or a single geometry. Moreover, some representation have level-of-detail and geometry simplification capabilities that help increasing the compactness. However, compact representations are also sensitive to errors and approximations since the scene is described with less data. Therefore, finding appropriate trade-off between compactness and accuracy is a challenging task for the design of a 3D video scene representation.

Compression compatibility. The compression of a representation corresponds to the number of bits needed to describe the scene at a certain quality. The compression of some types of representations like 2D images and videos is highly optimized since it has been studied for years, whereas some other representations, less popular or more recent, do not have dedicated and optimized compression methods. The compatibility with an already standardized compression method is an advantage for fast integration of the representation into a multi-view video system.

View-synthesis complexity. The view-synthesis complexity corresponds to the amount of processing needed to synthesize the views at the user side. It is highly related to the type of representation that is used, and more precisely to the type of rendering primitives employed. Real-time visualization is usually desired for interactive 3D video experience. Therefore, the view synthesis complexity should always be compatible with the performances of the display device. Thanks to recent graphics hardware and programming, most view-synthesis methods can run in real-time. However, if no powerful graphics hardware is available, e.g. on a mobile device, then a representation with low view-synthesis complexity must be used.

Navigation range and image quality. Each representation contains more or less accurate description about the scene and its geometry. Intuitively, a representation made of multiple textures and multiple geometries contains the local details of a scene and therefore provides higher navigation range and image quality than a representation made of a single texture and geometry. Artifacts in the synthesized image depend on the type of rendering primitive, they may be reduced by increasing the level of detail of the representation when possible. As already mentioned, finding appropriate trade-off between compactness and image quality is a challenging task for the design of a 3D video scene representation.

This section has defined some general properties of a representation and we have shown that the choice of a representation is of central importance in a multi-view video system. In the following sections, different types of representations are detailed and their properties are analyzed. The first one is the image-based representation.

3.2 Image-based representations

Image-based representations for multi-view video do not use geometry information at all. Only the color information or flow of light is transmitted (figure 3.2). In this section,

we first describe a system that uses the captured images directly as the representation. Second, we explain how the flow of light can be described from the input images using plenoptic modeling. Then, contributions that estimate a geometry of the scene at the user side are given. They enable to synthesize virtual views and thus reduce the number of original cameras. Finally, the standardized compression method for multi-view video is introduced.

Figure 3.2: An image-based representation made of four views.

A first 3D TV system (2004). In the work presented by Wojciech and Pfister [MP04], a 3DTV prototype system with real-time acquisition, transmission, and display is presented. Figure 3.3 gives an overview of this system. It consists of an array of 16 cameras arranged in a linear array, 8 producer PCs connected by gigabit Ethernet to 8 consumer PCs, and a multi-projector 3D display (figure 3.4). The video streams are encoded at full resolution (1300×1030) with MPEG-2 video codec and immediately decoded on the producer PCs.

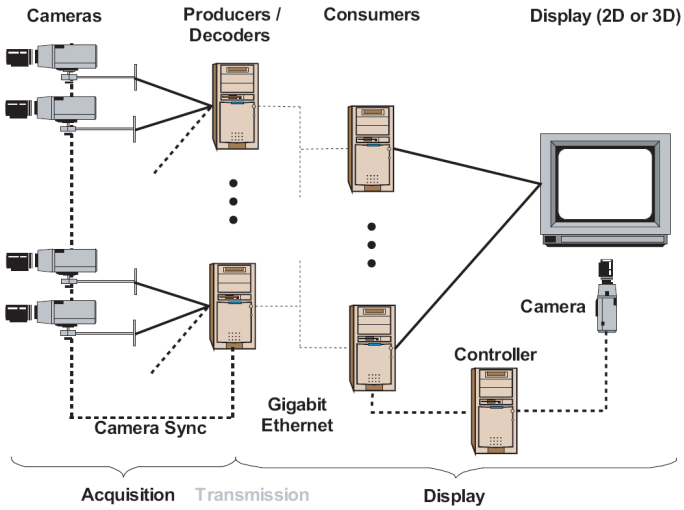
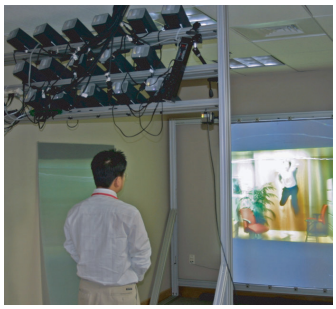


Figure 3.3: Overview of the 3D TV system [MP04]

The representation used here is image-based because the captured images are directly encoded and transmitted to the user. This system has the advantage of providing a "truly immersive 3D experience" because of its high number of cameras and projectors.

(a) Array of 16 cameras

(b) Array of 16 projectors



(c) Front-projection 3D display with single-lenticular screen

Figure 3.4: Acquisition and display system [MP04]

Moreover, all the data is processed in real-time which fits to an application scenario like live 3DTV broadcasting. However, transmission of all the data requires a very high bandwidth. Here, gigabit ethernet (1 Gbit/s) is used to connect producers and consumers PCs into a local area network. But transmitting such data load is not possible into a wide area network with lower bandwidth (usually range from 1 to 32 Mbit/s) like digital video broadcasting (DVB) or internet.

In a more recent work presented by Baker et al. [BL09], a similar system providing real-time capture and display of live multi-view video was set up: 9 cameras and 9 projectors are used and all the computations can be performed with a single PC. However, the same drawback as the previous approach remains: transmission of the all images still requires a high bandwidth.

Plenoptic modeling. Instead of storing all the scene information as pixel arrays, another method consists in describing the flow of light captured by the cameras. Such a flow is described by the plenoptic function. It was introduced by Adelson et al. [AB91]. It is characterized by seven dimensions, namely the viewing position (v_x, v_y, v_z), the viewing direction (θ, ϕ), the time t and the wavelength λ. Acquiring the full plenoptic function is not feasible because of the tremendous amount of data required. Therefore, research is mostly about how to make reasonable assumptions to reduce the sample size while keeping the rendering quality. Usually, the wavelength is reduced to red, green and blue components, and the time is sampled to a certain frame rate. One major strategy to reduce the data size is restraining the viewing space of the viewers.

The most well-known methods are Light Field [LH96], and the Lumigraph [GGSC96]: light rays are recorded by their intersections with two planes. One of the planes is in-

dexed with coordinate (u, v) and the other with coordinate (s, t). Figure 3.5 shows an example where the two planes, namely the camera plane and the focal plane, are parallel. A light ray is indexed as (u_0, v_0, s_0, t_0).

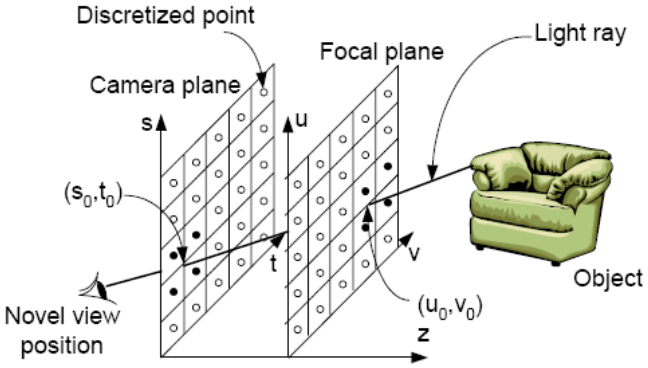


Figure 3.5: Illustration of the light-field parameterization. Image extracted from [ZC04].

During the rendering process, for each pixel i of the new view, a viewing ray r_i is computed that passes through the two-plane parameterization and generates a particular sample (u_i, v_i, s_i, t_i). If such a sample exists in the database, then the appropriate color value is assigned to the pixel i. If not, the nearest ones are selected and blended.

Another 4D re-parameterization of the plenoptic function is the Ray-Space [Fuj94]. This method is a combination of 4D light field and EPI (Epipolar-Plane Image analysis). Fuji and Tanimoto exploit this approach in their FTV system [Tan06].

All these methods using such image image-based representations suffer from the same limitations: a small-baseline camera arrangement must be used for good image quality because no geometry is available for virtual view synthesis. This results in either a high number of cameras if large navigation is desired or small navigation range if only few cameras are used.

Real-time geometry estimation. A way of improving the previous systems and reducing the bandwidth is to reduce the number of input cameras. However, to keep the same navigation range, this requires increasing the camera baselines (inter-camera distance) and virtual views between original cameras have to be synthesized using some geometry of the scene. Therefore, such solution estimates the geometry of the scene in real-time, at the user side of the system. In this case, we consider that the representation is still image-based because the geometry information is not transmitted but computed at the user side. Examples of real-time techniques compute depth maps from stereo images [YPYW04, TTN08, SKS+10], or reconstruct a 3D geometric model using visual hulls and photo-consistency constraints [MWTN04, NNT07]. In the work presented by Taguchi et al. [TTN08], a full 3D video system is presented. Acquisition

is done with an array of cameras and display is performed with real-time synthesis of virtual views. All these methods make use of the power of graphics hardware in order to obtain real-time performances. In the work of Sizintsev et al. [SKS+10], depth maps are estimated from a stereo pair at 32 fps on 640×480 video using off-the-shelf Nvidia GPU.

Although recent results show good quality geometry estimation with real-time capabilities, the main drawback of this solution is that it transfers computationally expensive tasks at the user side. This might be a critical point when considering a user application where computational capabilities are limited such as a mobile device or a television.

Compression of multi-view videos. A particularly challenging task for image-based representations is to efficiently compress the multiple views. Multi-view Video Coding (MVC) has gained significant attention recently. Since the different camera signals contain a large amount of statistical dependencies, the key for efficient MVC lies in the exploitation of these inter-view redundancies in addition to temporal redundancies: a frame from a certain camera can be predicted not only from temporally related frames from the same camera, but also from the frames of neighboring cameras. MVC is now an amendment to H.264/MPEG-4 AVC video compression standard developed with joint efforts by MPEG/VCEG [JTC08].

Pros and cons of image-based representations. The main advantage of image-based representations is a potentially high image quality. However, this benefit has to be paid by a high amount of data since a large amount of small-baseline cameras is required. In the case where high computational complexity at the user side is not an issue, then the number of cameras may be reduced and geometry computed in real-time at the user side.

3.3 Depth image-based representations

In depth image-based representations, depth maps are used together with the original 2D images in order to build a 3D-like representation. Such maps assign a depth value to each pixel of its associated image as shown in figure 3.6. They are used to synthesize virtual views thanks to so-called depth image-based rendering techniques (DIBR). In this section, we first show the different systems that can be envisioned with such representation and then detail the existing types of depth image-based representations (namely 2D+Z, MVD, and LDV) and associated compression methods. Finally important issues concerning the use of this representation for view synthesis are explained.

Example of systems. Currently, an ad hoc group of MPEG called 3DV is investigating a new framework using depth image-based representations and targets the compression and the generation of additional views at the receiver side [MPE09]. Figure 3.7 shows several scenarios of acquisitions and displays that share the common

Figure 3.6: Color image and its corresponding depth map [ZKU+04]

depth image-based representation. The depth maps may be produced by dense stereo analysis from multiple cameras[1] [SSZ01, Sou10]; with depth cameras or 2D/3D conversion processes [YXDL10]. At the receiver, depth image-based rendering is performed to synthesize images depending on the types of displays.

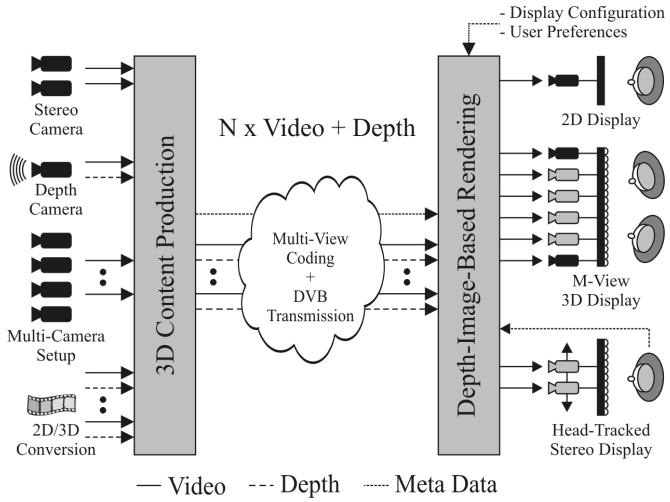


Figure 3.7: Example of multi-view video systems using depth image-based representations envisioned by the MPEG's 3DV group.

Video plus depth (2D+Z). The simplest depth image-based representation consists of using only one view made up of an image plus a depth map per time instant (2D+Z), as in figure 3.6. The ATTEST project [FKdB+02] was the first to study all parts of the 3D processing chain based on the 2D+Z representation. This representation was designed to replace a pair of stereo images for stereoscopic visualization. Figure 3.8 illustrates the display of a stereo pair of images using video plus depth and depth image-based rendering. In this illustration, the depth map is obtained using a depth camera, but it is also often computed by stereo analysis between two original views. There are mainly two advantages of using 2D+Z instead of stereo images. First, a depth map

[1] `http://vision.middlebury.edu/stereo/`

is a gray level image which is more compact than a 3 components RGB image and can be compressed more efficiently [BVG+07]. The compression of video plus depth has been standardized in the MPEG-C Part 3 specification [MPE06]. Second, a depth map allows to synthesize any intermediate views, so it provides flexibility for tuning the stereo visualization and decoupling the display side from the acquisition side. However, an image plus a depth map is not sufficient to fully reconstruct a second image. Indeed, some part of the scene may be occluded in the first image. These occluded regions are not well reconstructed during view synthesis and must be filled using some filtering methods [SMD+08] or inpainting methods [OYH09, Dar09]. These techniques exploit the neighboring pixels in order to predict the unknown ones, but only small holes can be filled without disturbing artifacts, thus only small distance between the original and synthesized views can be used.

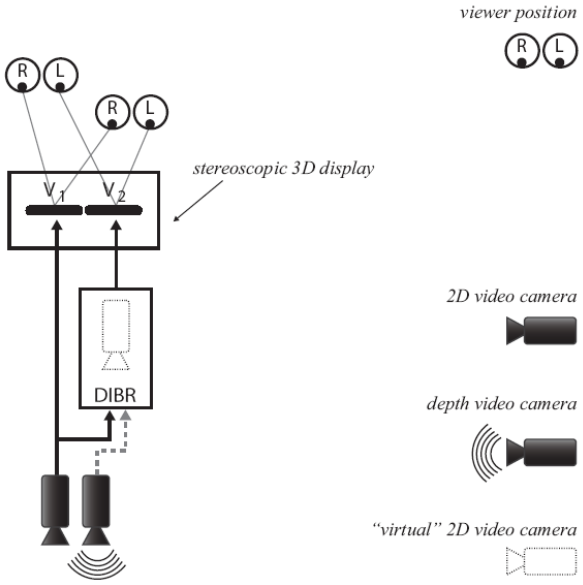


Figure 3.8: Display of a stereo pair of images using 2D+Z representation.

Multi-view video plus depth (MVD). The 2D+Z representation is only dedicated for stereo video (i.e. with small distance between the two images). For a wider range of viewpoints, multiple views made of 2D+Z data must be used. It is called MVD (Multi-view Video plus Depth) and enables to synthesize intermediate views with higher quality. Indeed, using MVD data, most of the occluded regions in one view can be filled using the other views [ZKU+04, SMD+08]. Usually, the two views surrounding a desired virtual view are used and combined together as shown in figure 3.9. Figure 3.10 illustrates the use of MVD for 3D video: three input images and associated depth maps are used to display nine views with a multi-view auto-stereoscopic display. On the down side, the data load is increased compared to 2D+Z and the redundancies between the original views are usually high since the same scene is captured by multiple cameras.

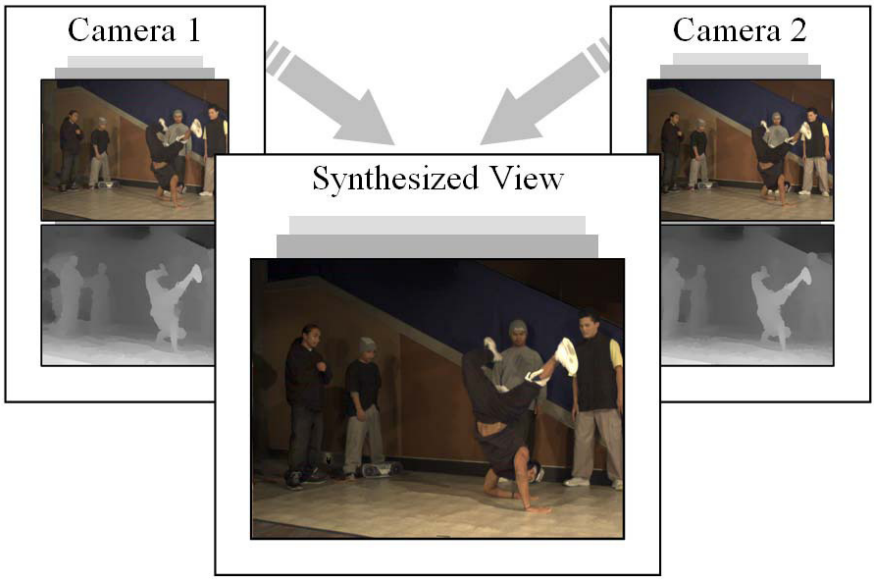


Figure 3.9: Synthesis of a virtual view using the MVD representation.[MSMW07b]

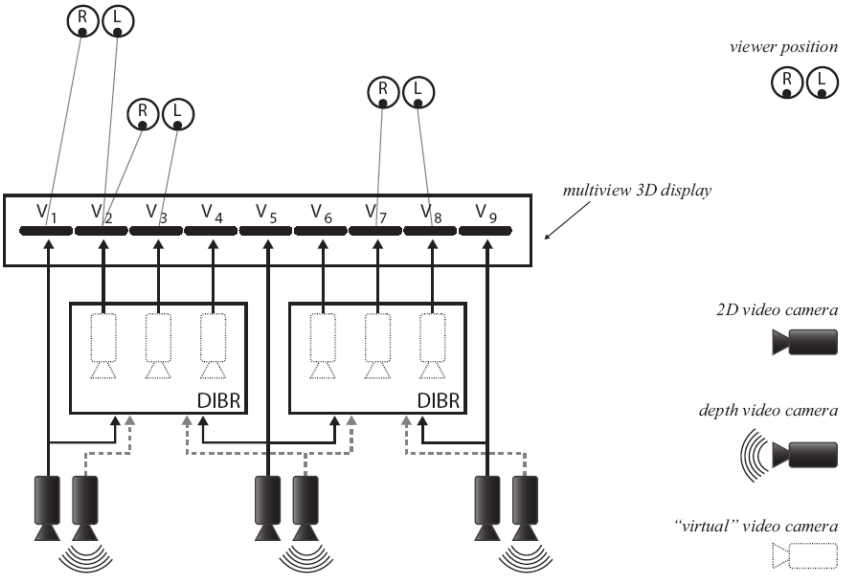


Figure 3.10: Display of multiple images using MVD representation.

Compressing MVD representation. The compression of depth maps and joint compression of multi-view plus depth is a very active research field. Since depth maps are gray level images, they can be compressed with already standardized video codec such as H.264/MVC as studied by Merkle et al. [MSMW07b]. Moreover, considering that the quality of synthesized views depends both on the quality of texture images and depth maps, then an optimized distribution of the bit rate over the texture and the depth helps minimizing distortions [Mor09, Dar09]. However, depth maps describe the surface of a scene and have different properties compared to an image describing the texture. Therefore, rendering intermediate views using compressed depth maps creates visually disturbing artifacts, especially around depth discontinuities (objects boundaries) [MSMW07b]. With this in mind, several approaches have been proposed such as platelet-based depth coding [MMS+08]. This algorithm employs a decomposition of the depth maps using geometric primitives such that the depth discontinuities are preserved. Wavelet based coding of depth map has also been proposed in order to preserve depth discontinuities [Dar09, MM09]. Another approach is the use of scalable coding technique with ROI to ensure lossless coding around depth discontinuities [YV09]. These methods provide a better rendering quality for a given compression rate.

Layered depth video (LDV). Starting from MVD representation, the LDV representation aims at reducing the multi-view redundancies while preserving important information like occluded regions. The idea is to select a certain view as reference and extract, from the other views, only the information which is not contained in the reference view, i.e. the occluded areas [SGHS98, BVG+07, MSD+08, JMG09]. Figure 3.11 illustrates this representation. The advantage is that the inter-view redundancies are reduced while the disocclusion areas are still available. However, the quality of synthesized views may decrease as the synthesis gets further to the reference view, on the contrary to MVD where all the original views ensure a certain view synthesis quality along the navigation range. Therefore, it is possible to combine both MVD and LDV ideas in order to play with the compromise between minimum quality and navigation range. This solution proposed by Smolic et al. [SMM+09] is called Depth Enhanced Stereo (DES). It uses two reference views, each associated with LDV representation. In between these views, the quality of synthesized views is equivalent to MVD, and for wider navigation on the right or left sides, the additional occlusion layers are used. Up to now, the compression of such LDV or DES representation is still under study [MDMW10, JMG09, KB10].

Depth based view synthesis Synthesizing intermediate views using depth maps is generally performed using a point-based method: each pixel is independently reconstructed in 3D and then re-projected into the desired intermediate view. As a result, many small holes appear in the intermediate view creating so-called sampling artifacts (figure 3.12). These holes must be filled with post-processing techniques [SMD+08]. An alternative to avoid these holes is to transform the depth maps into a surface using geometric primitives such as triangles [ZKU+04] or quadrilaterals [ESWK04] and to disconnect these primitives at depth discontinuities so that the background and

Figure 3.11: Layered depth video representation. Left: reference view and its associated depth map. Right: Occluded areas and associated depth maps extracted from MVD representation [MDMW10].

foreground are not connected. This solution eliminates the post-processing stage but requires a graphic processor for rendering these polygonal primitives.

Depth image-based representations suffer from a specific artifact when synthesizing virtual views called ghosting artifact (or corona artifact). Indeed, boundaries around depth discontinuities have mixed colors between foreground and background that become visible and disturbing after projection into another viewpoint (figure 3.12). Existing methods for avoiding this kind or artifacts consists in extracting discontinuity boundaries with an edge detection method, and applying separate process to this area [ZKU+04, SMD+08]. This process is performed at the user side of the system and therefore increases the view synthesis complexity.

Finally, for auto-stereoscopic visualization, images are often rectified, i.e. there is only horizontal parallax between the images. In this case, a simple method of virtual view synthesis is to use disparity maps instead of depth maps. Disparity is the difference in position between two corresponding pixels in two images. Thus, only linear interpolation of the disparity is needed for synthesizing a virtual view, instead of back-projection to 3D plus re-projection. This method is much less complex and is required by the MPEG 3DV working group [MPE09]. However only horizontal parallax is supported in this scenario, which is restrictive for free-view point functionalities and for future full parallax auto-stereoscopic displays.

Pros and cons of depth image-based representations. Depth image-based representations provide good flexibility to play with the trade-off between data load, image quality and navigation range. Moreover, compression of such representation is very active, but special attention must be given to depth discontinuities during the compression step. Moreover the view synthesis step requires many processings for avoiding specific artifacts.

(a) Sampling artifacts. (b) Ghosting artifacts.

Figure 3.12: Two examples of artifacts that require additional processings at the user side of the system. (a) Sampling artifacts are the small holes that create white lines in the image. (b) Ghosting artifact is the mixed color between the boundary of the head and the background.

3.4 Surface-based representations

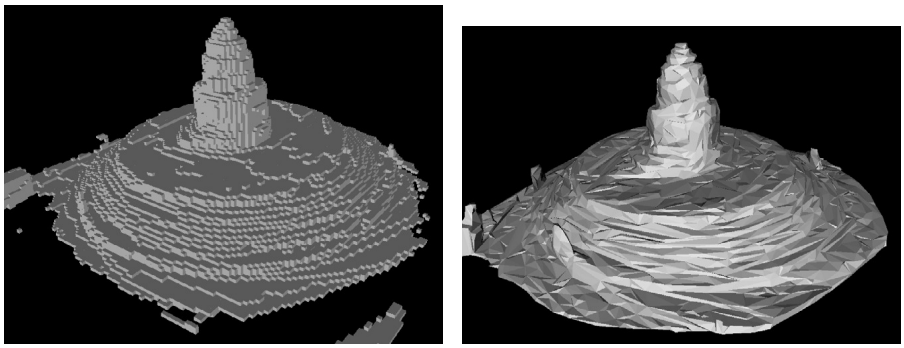
A surface-based representation models the surface of a scene using a certain type of geometric primitive. The most common primitives are the polygons and the points, thus two surface-based representations are described in the following: polygonal meshes and point-based surfaces.

3.4.1 Polygonal meshes

The surface of a 3D object can be approximated using polygons. A polygonal approximation of a 3D object has faces, edges, vertices and normal vectors to identify the spatial orientation of the polygon surfaces. These are stored in geometric data tables. There are several ways of constructing a polygonal mesh. First, a popular one is based on the shape-from-silhouette reconstruction algorithm, however it is restricted to foreground objects only. If the goal is to model a human body, then a priori knowledge can be introduced and a computer generated model adapted to the person's outline can be used. Finally, another solution is to reconstruct a polygonal mesh from several depth maps, thus complex scenes can be modeled without restriction concerning the scene. More details about these techniques are given in the following, and then compression issues are considered.

Mesh reconstruction from shape-from-silhouette In the work presented by Mueller et al. [MSM+04a], a shape-from-silhouette algorithm is used to reconstruct

a natural scene from multiple cameras. During silhouette segmentation, color-and-position-based automatic segmentation is applied first to create approximate segmentation that is refined manually afterwards. A hierarchical voxel approach is used in order to create an octree structure of the volume. If texture mapping is applied at this stage, visible artifacts would occur. Hence, the voxel model is transformed into a polygon mesh representation thanks to the marching cubes algorithm [LC87]. This general approach extracts the outer faces of voxel cubes that are part of the surface. The result is a very dense mesh of the object surface (figure 3.13). At the rendering stage, a view-dependent texture mapping algorithm is used.



(a) (b)

Figure 3.13: Transformation of a voxel model into smoothed wireframe [MSM+04a]

Introduction of a priori knowledge In the work presented by Carranza et al. [CTMS03], an a priori shape model that is adapted to the observed person's outline is employed. The model is a triangular mesh. On the contrary to shape-from-silhouette methods, using this kind of a priori model enables to prevent geometry artifacts. The body model used throughout the system is a generic model consisting of a hierarchic arrangement of 16 body segments (head, upper arm, torso, etc.). The model's kinematics are defined via an underlying skeleton consisting of 17 joints connecting bone segments (figure 3.14). The challenge in this method is to capture the motion of the body in order to reproduce this motion on the model. To do so, silhouette images of the person are extracted in each camera view through background subtraction. After an initialization step, the body pose parameters that maximize the overlap between projected model silhouettes and input camera silhouettes are estimated for every time step. More details and videos are available at the project's webpage[2].

Mesh reconstruction from depth maps In the work presented by Merrel et al. [MAW+07], a view-point based approach for the quick fusion of multiple stereo depth maps is presented. First, depth maps are computed from a set of images captured by a moving camera with known pose, using plane-sweeping stereo [Col96, GFM+07]. This

[2] `http://www.mpi-inf.mpg.de/~theobalt/FreeViewpointVideo/free_viewpoint_video.html`

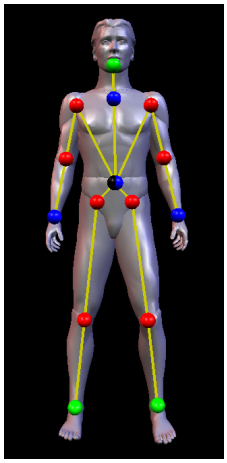


Figure 3.14: Surface model and the underlying skeletal structure [CTMS03].

algorithm generates potentially noisy, overlapping depth maps. Second, the depth maps from adjacent viewpoints are recursively merged by minimizing violations of visibility constraints. Two different approaches are presented, one that favors stability and one that is based on confidence. In both approaches, one of the viewpoints, typically the central one, is selected as the reference viewpoint, and a depth is estimated for each pixel of the reference view. The result is called a fused depth map, and several fused depth maps are computed throughout the video such that they partially overlap one another. The fused depth maps are then merged and converted to a consistent triangular surface with a multi-resolution quad-tree [Paj02]. The algorithm can run at up to 25 frames per second which makes it suitable for large scale reconstructions, as in the system proposed by Pollefeys et al. [PNF$^+$08].

In a similar idea, Galpin and Balter [Gal02, BGM06] estimate a set of local 3D models from a video sequence and estimated depth maps. A depth map is estimated for a group of frames, and it is then triangulated using Delaunay triangulation. This forms a local 3D model. One difficulty is that the vertices of two successive models are not matching points, whereas the models usually represent largely overlapping parts of the scene. Therefore, the transition between successive 3D models is achieved by using a single connectivity mesh that gathers the connectivity information and by morphing the 3D models to obtain a smooth transition.

On the contrary to the depth fusion in [MAW$^+$07] which is based on visibility constraints with regard to a reference view, the method in [BBH08] presents a point-based post-processing fusion. After the estimation of depth maps thanks to a stereo matching algorithm, they are all merged into a single point cloud, and the surface reconstruction algorithm runs in three steps: downscaling, cleaning and meshing (figure 3.15). During downscaling, the aim is to remove redundant information in the point cloud. The space is partitioned into a hybrid octree-quadtree structure and each subset of the tree is replaced by a single representative sample. During cleaning, two kinds of noise, namely 'outliers' and 'small scale high frequency noise' are removed thanks

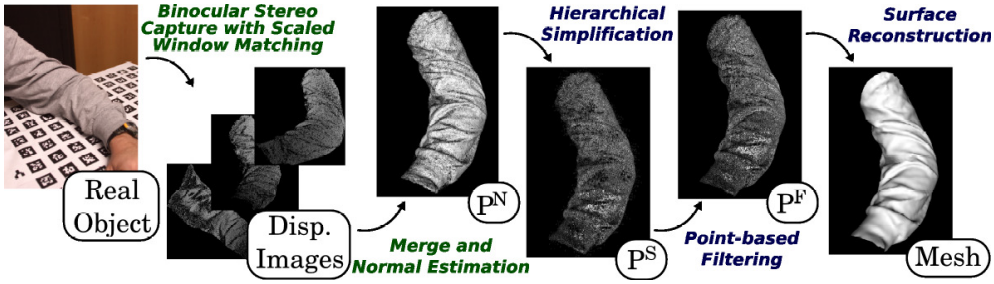


Figure 3.15: Acquisition pipeline: the binocular stereo algorithm generates a 3D point cloud that is subsequently processed and converted to a triangle mesh [BBH08].

to two types of filtering methods. Finally, the meshing step is done thanks to 'lower dimensional triangulation' methods which are fast and run locally, ensuring scalability and good memory-computational complexity. As a result, the authors claim to produce the most accurate results among current algorithms for a sparse number of viewpoints according to the Middlebury datasets. On the contrary to the previous algorithm, the authors make the assumption that a segmentation of the object from the background is provided, so that the visual hull is represented as a set of silhouette images. The algorithm is therefore suitable for foreground objects and not for large scale scenes.

Compactness and compression A polygonal mesh is a global model of the scene, thus there are no redundancies in the geometry resulting in a compact representation. Moreover, mesh reconstruction and simplification algorithms adapt the size and number of polygons to control the level-of-detail of the mesh and thus the trade-off between compactness and accuracy. In addition, since the number of polygons in a mesh may be huge, efficient compression methods are necessary. The compression of static meshes have been studied for long and more recently for dynamic scenes. The compression of polygonal meshes is studied within the MPEG's group 3DGC. Recent standards include the compression method TFAN [MZP09] for static meshes and FAMC [MZP08] for dynamic meshes, both part of the MPEG-4 AFX standard. More details about mesh compression can be found the works of Smolic et al. [SMS+07], Mamou [Mam08], Dugelay et al. [DBD08]. The compression methods for polygonal meshes are efficient for synthetic scenes or real-world foreground objects. However, arbitrary real-world scenes contain many depth discontinuities and irregularities that decrease the compression performances. In the work of Galpin and Balter mentioned above [Gal02, BGM06], efficient compression of real-world scenes is obtained by using a mesh that connects depth discontinuities (foreground and background are connected) resulting in strong artifacts when changing the viewpoint.

Pros and cons of polygon-based representations. Polygonal meshes are widely used in the computer graphics community, and therefore the available technology for rendering is optimized, enabling real-time view-synthesis of complex scenes. Moreover,

polygonal meshes are compact since the representation is made of a global model with controlled level-of-detail and without redundancies. On the down side, reconstructing a global model from multiple images or from the fusion of multiple depth maps remains error prone. More precisely, finding an optimal solution to fuse multiple and possibly inconsistent depth maps may introduce global errors. Considering polygonal meshes reconstructed with shape-from-silhouette techniques, they have the advantage to provide wide navigation range since surrounding cameras spaced far apart are often used, however only foreground objects of interest are reconstructed with this method. Finally, the compression of polygonal meshes of arbitrary real-world scenes is still a challenging task. Up to now, efficient compression method preserving depth discontinuities has not been proposed in the context of 3D video.

3.4.2 Point-based surfaces

Points can be used instead of polygons, as simpler display primitives for surface representation. In point-based schemes, no topology or connectivity information is explicitly stored. The points are represented by their 3D coordinates, their color and sometimes their normal for re-lighting effects. The idea of using points, instead of triangle meshes and textures, was first proposed by [LW85]. In the following, we first introduce the rendering method called splatting and then describe a full system using a point-based representation. Finally we discuss the compression methods of such representation.

Splatting as a rendering method. Most point-based rendering systems use splatting to achieve high quality rendering [PZvBG00, RL00, ZPvBG02]. The basic idea in splatting is to associate each surface point with an oriented tangential disc. The shape and size of the disc may vary, if it is circular, it projects as an elliptical splat on the image plane and its radius can be adjusted with respect to the local density. The shade or colour of the point is warped accordingly so that its intensity decays in the radial direction from the centre. Often a single image pixel is influenced by several overlapping splats; in this case the shade of the pixel is computed by the intensity-weighted average of the splat colors.

A full system using a point-based representation Waschbusch et al. [WWCG07], have studied a system based on a point-cloud representation. It aims at capturing and rendering 3D video sequences of real-world scenes. Figure 3.16 gives an overview of the 3D video framework.

The acquisition system is composed of multiple sparsely placed 3D video bricks that contain a low-cost projector, two greyscale cameras and a high-resolution colour camera. While the colour camera captures the texture of the scene, the remaining devices are used for the computation of the depth of the scene. Indeed, the proposed depth estimation algorithm makes use of stereo matching and structured light patterns that is projected onto the scene. The structured light pattern generates artificial textures that improve the stereo matching process. Therefore, alternating projections of structured

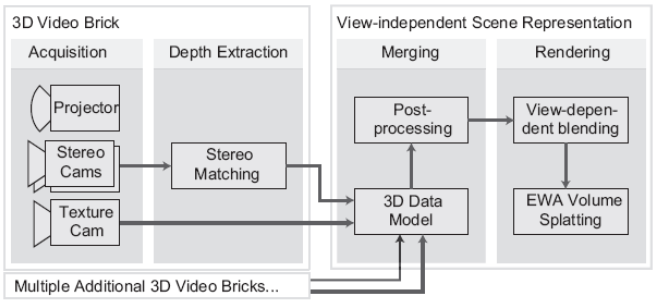


Figure 3.16: Overview of the 3D video framework [WWCG07]

light patterns and the corresponding inverses allows for simultaneous acquisition of the scene textures and geometry.

Once the depth maps have been estimated for each view, all the image pixels are back-projected into a common 3D world reference frame and they are merged into a view-independent, point-based 3D data structure. Then, the 3D points have to be represented as surface or volume elements in order to ensure full surface coverage of the samples. Hence, every point is modelled by a 3D Gaussian ellipsoid which corresponds to a probabilistic model describing the positional uncertainty of each point. Finally, the resulting point-cloud is post-processed to remove the remaining artefacts. At the rendering stage, the GPU and CPU are used cooperatively. Smooth images are generated using probabilistic EWA volume splatting and view-dependent blending. An example of a rendered image from a new view point is shown in figure 3.17, the quality is decent and could be improved by eliminating remaining artefacts at silhouettes using matting approaches.

Figure 3.17: Rendering from a novel viewpoint [WWCG07]

Compression While point-cloud representations show advantages at the construction and view synthesis stages for highly detailed surfaces sometimes made of millions of points, efficiently compressing such point-cloud is a challenging task in the context of 3D video system. Existing compression methods are often based on a subdivision of the space [DG00, PK03, ZXY07], but a simple prediction scheme was also proposed by Gumhold et al. [GKIS05]. The point-cloud representation has been included since 2004 into the MPEG-4 AFX standard which is a 3D graphics tools for geometry, texture and animation. However, only few experiments have been carried out to compress and transmit a point-cloud representation of a real world scene. In the work of Wuermlin et al. [WLG04], a point-cloud representation was used for a telecollaboration system called *blue-c* [GWN+03]. This 3D video system is able to acquire, process, transmit and render a model of an animated object at about 5 frames per second. However the system is restricted to foreground objects only, containing in between 15k and 25k points. In the work of Waschbusch et al. [WWCG07] (detailed above), the scene representation is adapted to arbitrary scenes and not to foreground objects only. However, the compression of their point-cloud representation is left as a perspective.

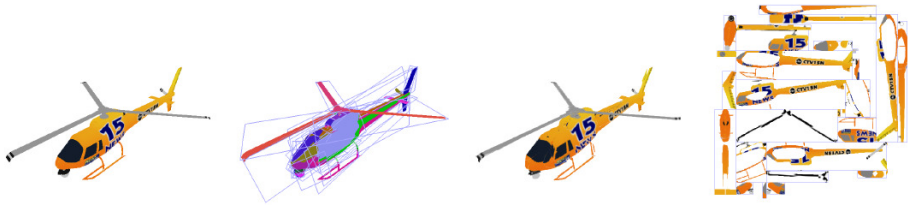
Pros and cons of surface-based representations. The main motivation for using points is the rendering complexity. Indeed, when a highly detailed complex 3-D triangle model is rendered, the projected size of individual triangles is often smaller than the size of a pixel in the screen in the image. In this case, the polygon rasterization process at the rendering pipeline becomes unnecessarily costly, whereas rendering individual points, rather than polygons, can be much more efficient. On the other hand, low-resolution approximations of point sets do not generally produce realistic rendering results. Therefore point-based representations are usually made of a huge amount of points which is a drawback when transmission with limited bandwidth is needed.

3.5 Impostor-based representations

Another class of representation is based on impostors. This term is used to describe a technique where flat images are seamlessly integrated into complex 3D scenes. Such impostor flat images are often called billboards to stress the idea that they face the camera similarly to billboards that are positioned to face drivers on a highway. In this section, we first introduce a method that models synthetic objects using billboard clouds. Then, we detail a system using billboard clouds for real-world scenes. Finally, a technique using smaller and oriented impostors called microfacets is described.

Billboard clouds for extreme model simplification. In [DDSD03], a billboarding technique is used to render complex geometric objects. This representation consists in a set of textured, partially transparent polygons (or billboards), with independent size, orientation and texture resolution (figure 3.18). A billboard cloud is built by choosing a set of planes that capture well the geometry of the input model, and by projecting the triangles onto these planes to compute the textures and transparency maps associated

with each plane of the billboard. Billboards clouds are effective in simplifying models with multiple textures into a small number of textured polygons.



(a) (b) (c) (d)

Figure 3.18: Example of a billboard cloud: (a) Original model (5,138 polygons) (b) false-color rendering using one color per billboard to show the faces that were grouped (c) View of the (automatically generated) 32 textured billboards (d) the billboards side by side. [DDSD03]

3D video billboard clouds. As an extension to the previous work, Waschbush et al. recently introduced 3D video billboard clouds [WWG07] where they reconstruct and represent a dynamic 3D scene using displacement mapped billboards. It consists in geometric proxy planes augmented with detailed displacement maps (figure 3.19). Since the displacement maps are computed using acquired depth maps of the scene, the representation used in this system is in fact not purely impostor-based but a combination of impostor and depth maps representation.

(a) (b) (c)

Figure 3.19: Illustration of the billboard cloud for one object: Billboard plane from one input view (left), composition of planes from multiple input views to a billboard cloud (middle), displacement-mapped billboard plane from one input view (right). [WWG07]

Their acquisition system consists of so-called 3D video bricks and the disparity maps are computed using stereo on structured light algorithm slightly different to the one described in [WWCG07] section 3.4.2. Alpha mattes are computed in order to segment the scene into multiple distinct objects. Then, for each object a billboard cloud is reconstructed consisting in one textured billboard plane per input viewpoint. Each

plane provides a least squares optimal approximation of the object geometry from its particular acquisition view. During rendering, view-dependent geometry and texture are computed by blending pixels depth and colours. However, the planar approximation is not sufficient for decent visual quality. Therefore, displacement map is added using input depth. Since the reconstructed geometry is subject to noise and outliers, a bilateral low-pass filter is employed to smooth the object geometry in space and time. In order to model complete dynamic scenes into multiple 3D video billboard clouds, a semi-automatic video cutout technique is used. Finally the moving scene can be re-rendered from novel viewpoints of a virtual camera.

A video of this method can be watched at the project's webpage[3]. The reconstructed scene shows good visual quality and coherence in time with only 3 or 4 input views. However, a limitation comes from thin and fast moving structures in the video that are difficult to handle (figure 3.19). The proposed method is in principle capable to run in real-time because efficient GPU implementations for both window-based and splatting are available.

As a future work, the authors would like to look for efficient compression method for this representation.

Figure 3.20: Thin, fast moving structures like the actor's arms cannot provide sufficient spatio-temporal support for high quality filtering. [WWG07]

Microfacet billboarding. Yamazaki et al. [YSK+02] proposed a new impostor-based representation in order to render complex objects in real-time. The billboards here are replaced by view-dependent microfacets. These microfacets are extracted from a voxel model of the scene and their orientation always faces the viewing direction (figure 3.21). This is a particularity of this representation: while other representations are made of a single geometry or multiple geometries, here microfacet billboarding is made of single view-dependent geometry. The geometry of the object is acquired thanks to a scanner and the texture is obtained with 36 photographs. The whole process can be separated into two steps: the modeling and the rendering process.

In the modeling process, the surface of the object is built in the form of a polygonal mesh and is then resampled into a set of voxels. This voxel representation can be extended to a multi-resolution structure by using octree representation if necessary.

[3] `http://graphics.ethz.ch/publications/papers.php`

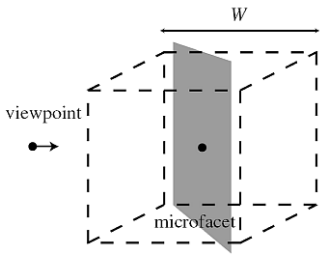
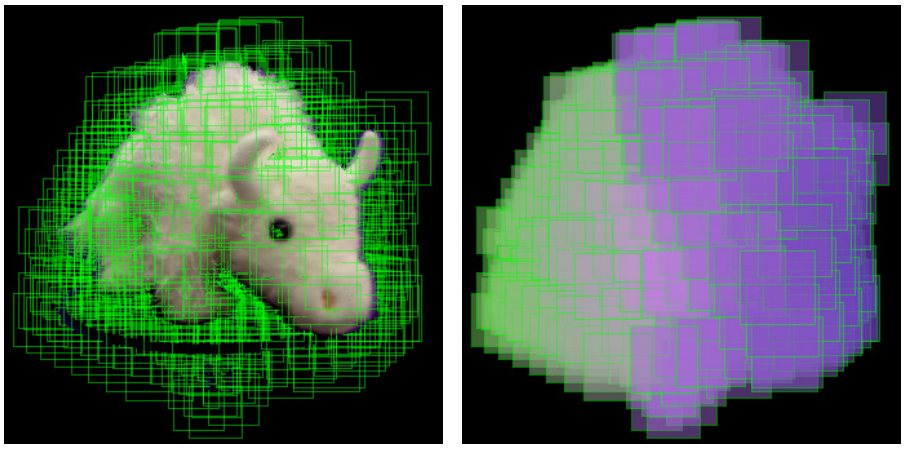


Figure 3.21: A microfacet is a slice which intersects the center of the voxel and always faces the viewing direction. [YSK+02]

In parallel, the range images are stored for texture clipping in the final step of the rendering process.

In the rendering process, the object is rendered by a set of microfacets with color texture. A microfacet is defined as a slice which intersects the center of the voxel and faces the viewing direction. Each microfacet represents the approximated surface inside the voxel (figure 3.21). First, view-dependent microfacets are generated, and then the texture of the object is mapped onto each of them (figure 3.22 (a)). The texture of each facet is selected from the most suitable texture images according to the view point. Finally, the background texture is removed from the facets either by texture clipping using the range image or by alpha matting.

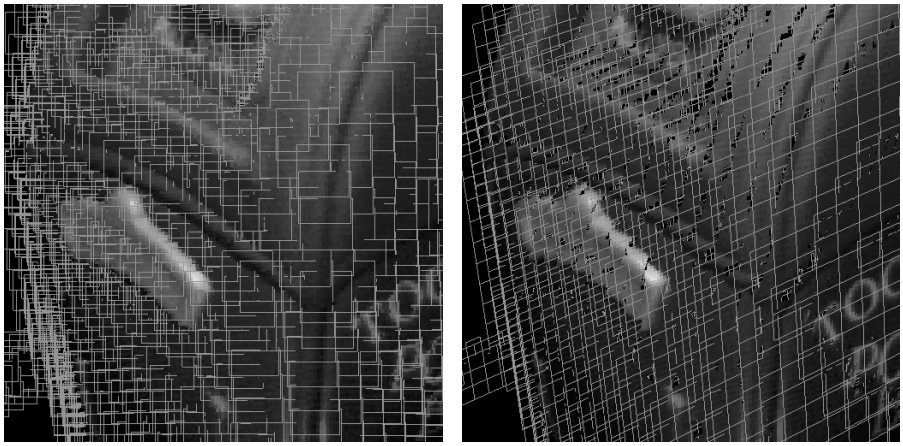


(a) (b)

Figure 3.22: Images obtained by microfacet billboarding rendering of a stuffed cow. (a) The result of rendering by microfacet billboarding using 756 facets. (b) The textures mapped onto facets are generated by interpolation of several textures. [YSK+02]

Since facets are used to represent real objects with small geometries, visual artifacts can occur depending on size, shape and orientation of the microfacets. Therefore, the authors studied the visual artifacts due to the sampling interval and orientation of the

microfacets. In figure 3.23, a comparison of the rendered images using billboarding facets and fixed facets is shown.



(a) (b)

Figure 3.23: Comparison of the rendered images using billboarding facets (a) and fixed facets (b). The same number of facets were drawn in each figure. [YSK+02]

The octree structure is useful to control the extent of approximation and the levels of detail. When the shape of an object is represented by a single voxel and approximated by a single microfacet, the method is equivalent to image-based representation. Whereas when the octree is subdivided into the size of the points in the original point cloud, the rendering becomes closer to point-based representation.

A video of this method can be watched at the project's webpage[4]

Pros and cons of impostor-based representations. Such representation helps simplifying the geometry by using billboards of different size without connectivity, thus real-time rendering of complex scenes is facilitated. Concerning billboard clouds, extracting large billboards from real-world scenes is difficult because of the complexity of shapes, resulting in poor approximation of the scene. Therefore, depth maps are used for displacement mapping but this increases the data load and thus the advantage of the compact billboard cloud is lost. On the other hand, microfacet billboarding is more flexible since the size of the microfacets can be adapted to the desired precision using the octree structure, and the orientation of the facets is view-dependent for a better rendering quality. Moreover, the octree structure can be exploited for compression. On the down side, the texture clipping and transparency inside each microfacet has to be defined during view synthesis. This is done either by using the depth maps if available (but this again increases the data load) or by alpha matting which is computationally expensive.

[4] `http://www.dh.aist.go.jp/~shun/research/microfacet/`

3.6 Summary and analysis of pros and cons

In this chapter, four families of representations for multi-view video have been presented, namely image-based, depth image-based, surface-based and impostor-based representations. Some differences were also distinguished inside each family. Table 3.1 regroups these representations and the references associated to them.

Family	**Representation**	**References**
Image-based		
	Image-based	[MP04, BL09, TTN08, AB91, LH96, GGSC96, Fuj94, Tan06]
Depth image-based		
	MVD	[FKdB+02, ZKU+04, SMD+08, SMM+09]
	LDV	[SGHS98, BVG+07, MSD+08, JMG09, SMM+09]
Surface-based		
	Polygon mesh	[MWTN04, MSM+04b, MSM+04a, CTMS03, BBH08, MAW+07]
	Point cloud	[LW85, PZvBG00, RL00, ZPvBG02, WWCG07]
Impostor-based		
	Billboard cloud	[DDSD03, WWG07]
	Microfacet billboarding	[YSK+02]

Table 3.1: Summary of representations and associated references

Before starting the analysis, few precisions about the representations that will be compared are necessary:

- One constraint is that the computational complexity should be transfered at the operator side as much as possible, in order to reduce complexity at the user side. Therefore, image-based representations will be considered without computation of the geometry at the user side, which is a complex task that should be performed at the operator side.

- Surface-based representations reconstructed with visual hull techniques are usually applied for foreground objects of interest only, which does not satisfy the constraint of arbitrary scenes defined for the system. Therefore, the analysis will focus on surface-based representation reconstructed from depth maps such that any arbitrary scene is possible. Billboard clouds and microfacets billboarding are reconstructed from depth maps and can be applied on arbitrary scenes, although the mentioned contributions focused on single objects.
- Billboard clouds without displacement mapping (i.e. without use of depth maps) will be considered in this analysis so that to clearly distinguish this representation from the depth image-based representations.

The processing steps associated to each representation are summarized in table 3.2.

	Construction	**Compression**	**View-synthesis**
Image-based	Color correction	Multi-view video coding (H.264/MVC)	Image interpolation or light field rendering
MVD	Depth estimation	Block-based coding (e.g. MVC) + edge preservation	Depth warping + post-processing
LDV	Depth estimation + layer extraction	Block-based coding + edge preservation + layers coding	Depth warping + post-processing
Polygon mesh	Depth estimation + fusion + meshing	Mesh-based object compression (e.g TFAN, FAMC)	Polygon warping + texture mapping
Point cloud	Depth estimation	Prediction in subdivided 3D space (e.g. octree)	Surface splatting
Billboard cloud	Depth estimation + billboard approx + texture clipping		Polygon warping + texture mapping
Microfacets	Depth estimation + Octree partitioning	Prediction with octree structure	View-dependent polygon warping + texture clipping and mapping

Table 3.2: Summary of the processing steps for representation.

In the following of this section, the representations are compared to each other

according to each property and also according to the constraints on the targeted system that we have defined in section 2.6. Finally, a global conclusion will extract the important observations that come out from this study.

3.6.1 Construction complexity

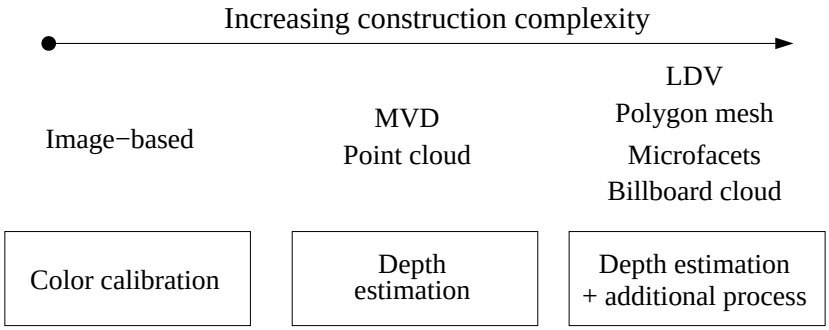


Figure 3.24: Representations ordered by increasing construction complexity

Figure 3.24 gives a classification of the representations by increasing construction complexity. Since image-based representations do not reconstruct the geometry of the scene, only some image processing algorithms may be needed such as color calibration. Therefore this representation is constructed with low computational complexity. Reconstruction methods based on depth estimation are more complex because of the pixel-by-pixel estimation. Finally, the other representations require additional process after depth estimation. LDV representations compute their layers from MVD. Polygonal meshes fuse depth maps and triangulate point cloud data. Billboards approximate the geometry from each viewpoints. Microfacets require the transformation of depth maps into a voxel representation. All these additional processes increase the construction complexity.

3.6.2 Compactness

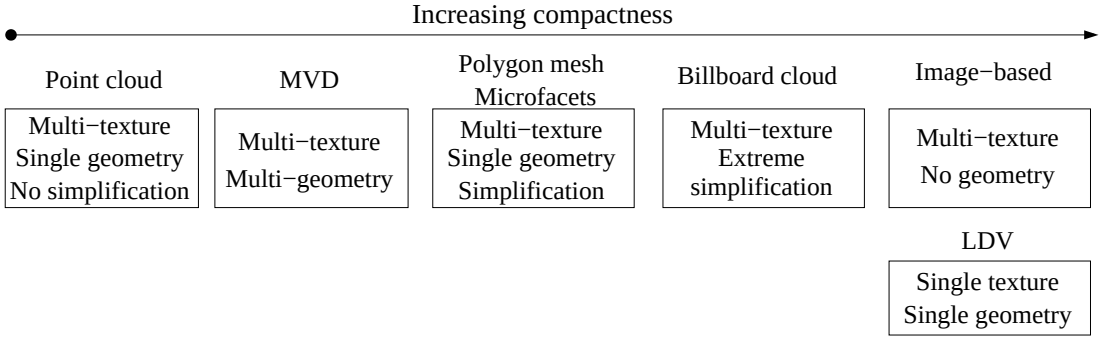


Figure 3.25: Representations ordered by increasing compactness

Figure 3.25 gives a classification by increasing compactness. Although the point-cloud representation is a non redundant single geometry, the number of 3D points is often huge because it is mainly adapted to describe highly detailed scenes. Simplifying a point-cloud is difficult since a single color is associated to each point and so lower approximation does not produce realistic results. Therefore, the point-cloud representation is not compact. The MVD representation uses multiple depth maps that require less data than 3D points because only gray scale component is used. However, they contain many redundancies. Therefore MVD is situated next to point-cloud representations. Polygonal meshes and microfacets representations are both based on a single geometry and polygonal primitives. Such primitives, allow geometry simplification to a certain extent, therefore these representations are more compact than the two previous. The billboard cloud representations uses few planes per view which results in a simple and compact geometry, but also extreme simplification that roughly models the scene's geometry. On the contrary to previous representations, LDV representations is based a single texture and single geometry. All redundancies are eliminated which makes this representation very compact. Finally, image-based representation do not use geometry information at all, which makes it also very compact when compared with the other representations with the same camera arrangement and input views.

3.6.3 Compression compatibility

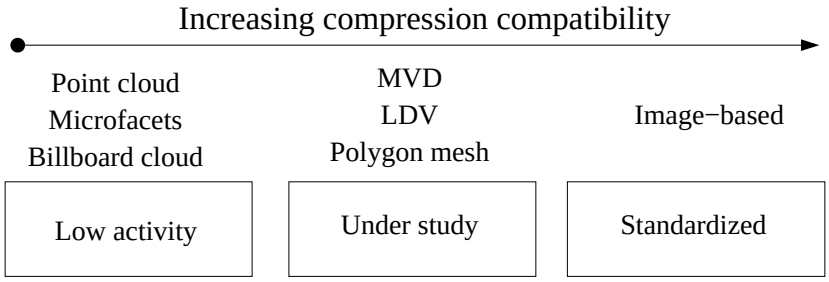


Figure 3.26: Representations ordered by increasing compression activity and standardization

Figure 3.26 gives a classification of the representations by increasing compression activity and standardization. Point-cloud and microfacets representations are situated in the 'low activity' class since only few experiments on real-world data have been carried out to compress such representations, and no recent contributions show that point-cloud or microfacets can be efficiently compressed in the context of 3D video. Billboard cloud representations are made of a few planar polygons only. However, it seems that the compression of such representation in the context of 3D video has not been studied a lot up to now. The 3D video group within the normalization group ISO-MPEG is currently studying the MVD and LDV representations. The compression of depth maps and joint compression of multi-view plus depth is a very active research field. The compression of polygonal meshes is studied within the MPEG 3DGC group

and compression standards already exists (such as TFAN and FAMC). However, mesh compression methods seems to be more efficient for a single foreground object than for a scene with arbitrary discontinuities. Therefore, the MVD, LDV and polygon mesh representations are situated in the 'under study' class on the middle of the axis. Finally, image-based representation has its own standardized compression method (MVC) and continuous improvements for image compression are studied within the MPEG and VCEG groups.

3.6.4 View synthesis complexity

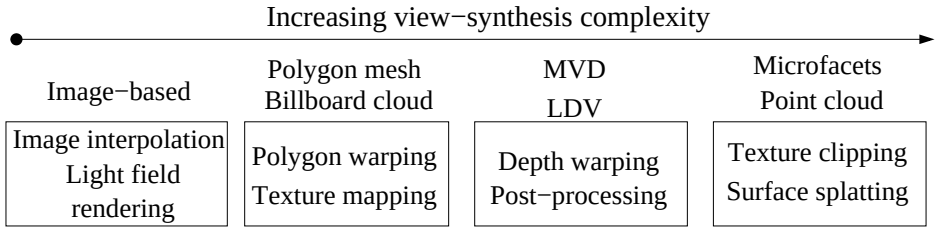


Figure 3.27: Representations ordered by increasing view synthesis complexity

Figure 3.27 gives a classification of the representations by increasing complexity of their view synthesis method. Image-based representations have a low view synthesis complexity since only image interpolation or light field rendering is done, without any geometry information. Then polygon mesh and billboard cloud representations are made of polygonal primitives and thus benefit from the high optimization of graphics hardware for polygon warping and texture mapping. MVD and LDV representations require pixel-by-pixel depth warping and additional processes like edge detection and median filtering for removing sampling artifacts and ghosting artifacts [SMD+08]. Thus it is considered more complex than the two previous representations. Finally, microfacets need the computation of texture clipping in every microfacet, and point clouds render a huge number of points with surface splatting techniques. These last two requires many computations and are therefore on the right of the complexity axis.

3.6.5 Navigation range and image quality

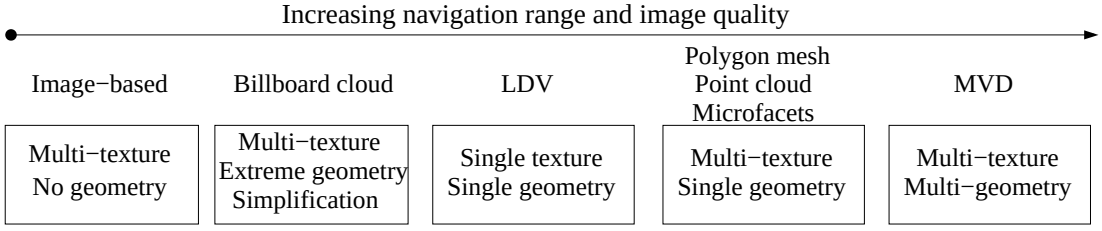


Figure 3.28: Representations ordered by increasing navigation range and image quality.

Figure 3.28 gives a classification of the representations by increasing navigation range and image quality. Synthesizing intermediate views using image-based representations does not compensate the parallax effect due to objects of different depth in the scene, therefore only small navigation is possible otherwise poor quality image would be obtained. Adding some geometry about the scene helps increasing the navigation range. Billboard cloud representations use an extreme simplification of the geometry scene leading to artifacts in detailed areas of the scene. Therefore it is situated close to image-based representation. Representations having single texture and geometry such as LDV suffer from global approximation and static appearance of the scene: reflectance, illumination changes and geometry inaccuracies can not be compensated. Therefore, LDV representations provide average navigation and image quality. Then representations using multiple textures and single geometry result in higher image quality than the LDV representation since view-dependent details are given. The three representations polygon mesh, point cloud and microfacets can adapt the size and number of their primitives to the scene geometry. This allows to play with the trade-off between compactness and image quality. Inaccuracies due to the single geometry can be compensated by multi-texturing methods such as floating textures [EDS+08], but this greatly increase the complexity of the view synthesis process. Finally, representations that use both multiple textures and geometries are capable of very good quality images. The MVD representation lies in this category. Depth-based representations like MVD and LDV suffer from sampling artifacts at the view synthesis stage which requires additional filtering process. On the contrary, surface elements such as polygons or splats take into account the continuity of the surface.

3.6.6 Summary of pros and cons

This section sumarizes the pros and cons of each representation. They are given in table 3.3.

Finally, a diagram is given in figure 3.29 to highlight the pros and cons of the representations. In order to simplify the visualization of this diagram, only some elements from this study have been included: the most popular representations for 3D video have been kept, namely image-based, MVD, LDV and polygon mesh. Moreover, the construction complexity is not included in the diagram since no constraints about it has been defined in our targeted application. Moreover, the view synthesis complexity property is changed as view synthesis simplicity such that the same interpretation can be done for each axis: the end of the axis means high advantage and the start of the axis (center) is low advantage.

This diagram is useful to see that each representation exhibit advantages and drawbacks with regard to our targeted system and that finding a suitable representation requires playing with many trade-offs. If only the advantages of each representations are kept, then a perfect representation would:

- be compact by removing redundant information (such as LDV)
- be compatible with an efficient compression method that avoids strong artifacts

Representation	**Pros**	**Cons**
Image-based	Compact; standardized compression method; Low construction and view synthesis complexity	Small navigation range and image quality
MVD	high navigation range and image quality; Compression under study	Not compact; Complex view-synthesis to avoid artifacts
LDV	Compact; Compression under study	medium navigation range and image quality; Complex view-synthesis to avoid artifacts
Polygon mesh	Compact; medium view-synthesis complexity	Compression more efficient for single objects than arbitrary scenes
Point cloud	Adapted to highly detailed scenes	No geometry simplification (dense point cloud); Complex view synthesis
Billboard cloud	Compact (Extreme simplification)	Low navigation range and image quality
Microfacet billboarding	Compact	Complex view synthesis

Table 3.3: Table highlighting the main pros and cons of the representations.

(such as image-based)

- avoid complex view synthesis stage (like image-based without geometry, or polygon-based with geometry)
- provide good navigation range and image quality using multiple textures and geometries (such as MVD)

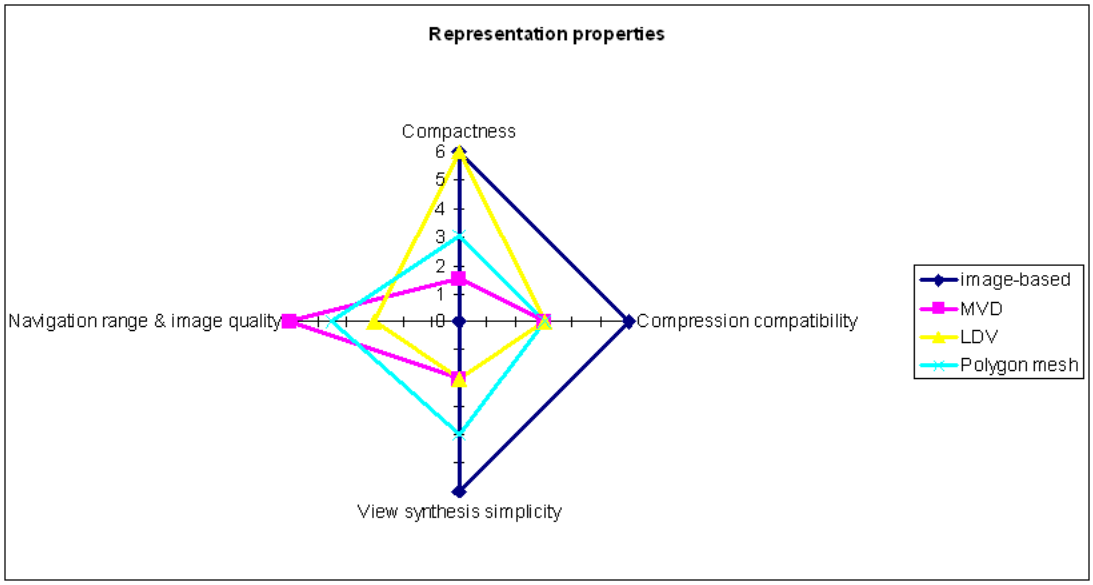


Figure 3.29: Pros and cons of most popular representations.

3.6.7 Conclusion

A few important conclusions come out from this study:

Applications. Each representation has its own advantages and drawbacks which may be more adapted to some applications than others. For example, image-based representations seems suitable for stereoscopic visualization since only two original views are needed. However, if more views or some navigation is desired then the image-based representation would not be adapted. Representations using a geometric approximation offer a wider range of view points and are then more suitable for multi-view autostereoscopic visualization or free-view point navigation. No existing representation has yet proved to be fully suitable for such applications. The representation that raises most interest is the MVD one because of its navigation range and possible compression compatibility with existing standards. However this representation is not compact and its compression is still under study. Therefore, there is a need for representation and compression for 3D video applications.

Geometry information. The geometry information allows synthesizing virtual views inside the viewing area. This feature is very important since it makes the representation flexible in many aspects. First, an arbitrary number of views can be synthesized at the rendering stage; therefore the representation is compatible with many different display devices, even if they don't display the same configuration of views. Second, the camera set-up of the acquisition system is also very flexible since arbitrary viewpoints can then be generated. This allows separating the acquisition configuration from the display constraints. But still, the camera set-up has to be precisely calibrated and the camera baselines should be small enough to allow good quality virtual views. Finally, geometry information is used for synthesizing virtual views instead of storing all the required views, thus less data storage is needed. Therefore, a suitable representation for 3D video should contain geometry information.

Polygonal primitives. Polygonal primitives have several advantages. For compactness, the polygon's size can be adapted to the geometry so as to keep the number of polygons low. Then at the rendering stage, polygons do not create cracks or sampling holes because they model the continuity of the surface. On the contrary, depth warping techniques require additional filtering process to fill these cracks that appear between the points. Moreover, real-time rendering can be achieved using graphics hardware and programming. Therefore, a polygon-based representation may be well adapted to 3D video applications.

Compactness VS quality. This study has shown that representations having multiple textures or geometries provide good quality images at the expense of the compactness, which is a classical trade-off. A more compact representation is usually obtained by using a single texture/geometry, but it is error prone and therefore the image quality is lowered. Typically, the MVD and LDV representations are two extremes: the first one keeps all the redundancies whereas the second one removes all of them. No representation address simultaneously the problem of compactness and quality at a local level, i.e. which areas of the scene should have multiple texture/geometry (for quality) and which areas should have only one (for compactness).

Single VS multiple texture/geometries. Apart from the compactness considerations, using either single texture/geometry or multiple ones both have drawbacks for synthesizing virtual views. Indeed, the issue when using a single occurrence is that it must be accurate enough to provide good images for the whole navigation range. In this study, we have seen that fusing multiple depth maps into a single model is a challenging task. On the other hand, using multiple texture/geometries raises the problem of consistency between these multiple occurrences and how to combine them when projected into a virtual view. Small inconsistencies between views sometimes create visible artifacts in virtual views. Therefore, both single and multiple texture/geometries have drawbacks for synthesizing virtual views. At this point, we want to stress the difference between multiple and view-dependent texture/geometries: the first is a combination

of multiple occurrences, the second is a transformation of the data. It is possible to combine the two: multiple view-dependent geometries. The microfacet representation is based on multiple textures and a single view-dependent geometry: small planar rectangles are oriented to face the current view-point. However, since a single geometry is used and structured in an octree, reconstruction errors are not corrected by the orientation of the microfacets.

Compression. The compression of the representation is important for bit rate reduction and image quality. The image-based representation can be compressed with the existing multi-view coding standard (MVC) and the polygon mesh representation can be compressed with the 3D mesh compression standard (TFAN or FAMC). However, the representation has limited navigation range and the second is mainly designed for single object rather than arbitrary scenes. Since depth maps allow synthesizing multiple-viewpoints with arbitrary scene content, many studies are driven so that to compress these depth maps with block-based coding techniques. The main problem up to now is to take special care of the depth discontinuities and to find the appropriate bit rate allocation between color and depth data.

Chapter 4

Overview of the proposed representation

Contents

The goal of this chapter and the following ones is now to present a new representation for multi-view video according to the conclusions of the previous chapter about existing representations. The main point for proposing a new representation is that it must take into account in a unified manner all the processing stages and constraints of the targeted multi-view video system. This chapter gives an overview of the proposed representation. First, we detail the input data that are used to validate the representation. Then we describe the proposed representation and analyze its properties. Finally, the different processing steps related to the representation are resumed.

4.1 Input data

Depth maps often serve as a starting point to geometric modeling since it provides depth information for each pixel and each viewpoint. This information can be further processed to result in any other geometry representation such as a point-cloud or a polygon mesh. In the following, one depth map per input viewpoint is supposed to be available. These depth maps may be acquired with depth sensors or estimated with a stereo algorithm. In addition, camera calibration parameters as well as depth near and far values are needed. An example of multi-view video plus depth input data is shown in figure 4.1 with the sequence *Breakdancers* and *Ballet*. They have been acquired

by Microsoft [1]. Eight views, resolution 1024×768, arranged on a horizontal arc span about 30° of the scene. In the figure, three of the eight views corresponding to cameras 1, 3 and 5 span about 17° of the scene. The available intermediate views 2 and 4 can be used for evaluation of the quality of the synthesized virtual views (see figure 4.3). Another MVD sequence called *Book Arrival*[2] is illustrated in figure 4.2. The resolution is also 1024×768. The camera configuration with this last data set is different: the cameras are rectified and the baseline is much shorter because 16 cameras span about 18.5°. For experimentations, three views corresponding to cameras 6, 8 and 10 spanning about 5° will be used. Intermediate cameras 7 and 9 will be used for evaluation of the synthesized views.

4.2 The polygon soup representation

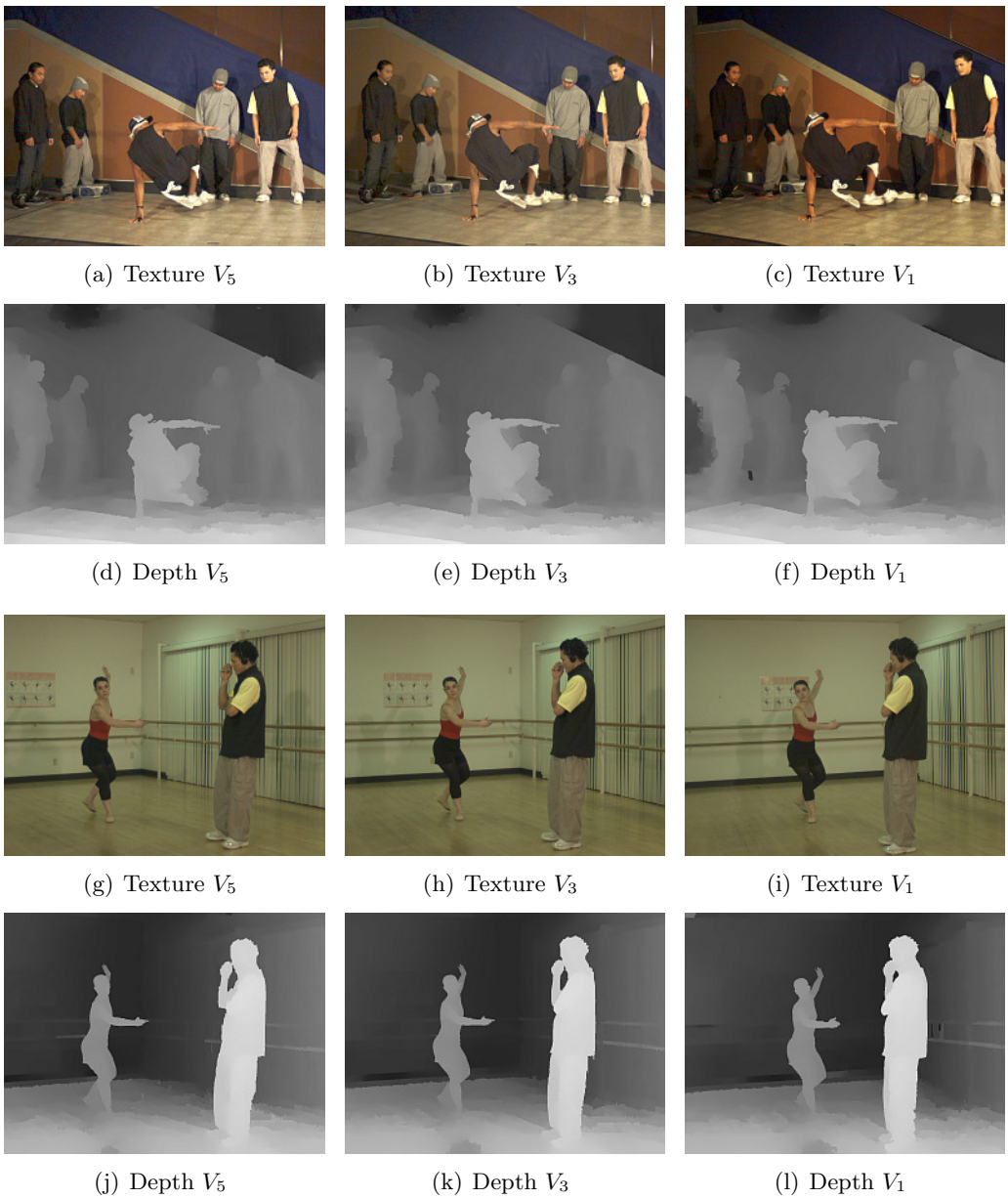
3D polygons. We propose to construct, from the input MVD data, a representation called polygon soup (figure 4.4). First of all, the rendering primitives are 3D polygons. As seen in the previous chapter polygonal primitives have several advantages: they increase the compactness, model the continuity of the surface, and graphics processors are optimized to render them. Moreover, all the 3D polygons are not necessarily connected to each others and can overlap, forming a kind of mixture of polygons which is often called a polygon soup. Here, the disconnection feature is very important. First, it ensures that foreground and background objects are not connected, thus preserving depth discontinuities. Second, it allows to easily remove redundant or unreliable polygons, thus increasing the compactness. The overlapping feature is also a key point. Since multi-texture and multi-geometry are available from the input data, overlapping polygons (i.e. coming from different views) can be selected or merged depending on the desired view-point, thus ensuring view-dependent quality of virtual views. In a word, disconnection and overlapping allow to play with the compactness and image quality trade-off by removing unnecessary polygons and overlapping necessary ones.

Stored in 2D. The representation forms a 3D polygon soup at the view synthesis stage. However, for compactness and compression efficiency, the polygons are stored in 2D with depth values at each corners. These 2D polygons are extracted from the depth maps using a quadtree decomposition method. In the following, we call the 2D polygons 'quads'. Figure 4.4 illustrates the relation between the 3D polygon soup and the quads extracted from the depth maps. As can be seen, one quadtree is extracted per depth map.

The decomposition of the depth maps into quadtree allows to retrieve the (x, y) positions of the quads, thus a compression method exploiting this structure can be employed. Figure 4.5 illustrates a quadtree decomposition and corresponding structure. Each leaf of the tree corresponds to a quad. Using this structure, it is easy to re-

[1] *Thanks to the Interactive Visual Media Group of Microsoft Research for providing the data sets*

[2] Thanks to the the Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute for providing the data.



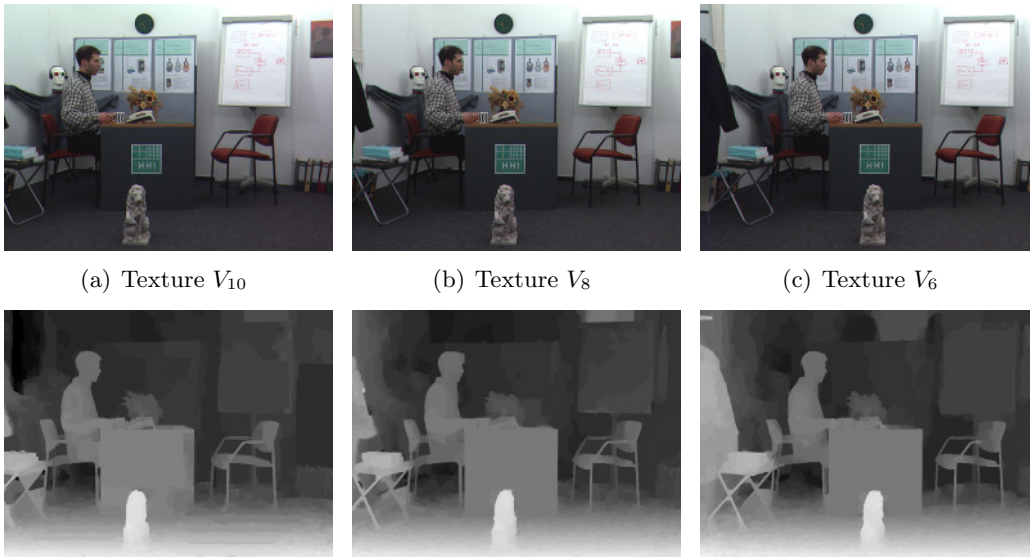
(a) Texture V_5 (b) Texture V_3 (c) Texture V_1

(d) Depth V_5 (e) Depth V_3 (f) Depth V_1

(g) Texture V_5 (h) Texture V_3 (i) Texture V_1

(j) Depth V_5 (k) Depth V_3 (l) Depth V_1

Figure 4.1: Example of viewpoints 1, 3, 5 and associated depth maps of *Breakdancers* and *Ballet* video sequences.



(a) Texture V_{10} (b) Texture V_8 (c) Texture V_6

(d) Depth V_{10} (e) Depth V_8 (f) Depth V_6

Figure 4.2: Example of viewpoints 6, 8, 10 and associated depth maps of *Book Arrival* video sequence.

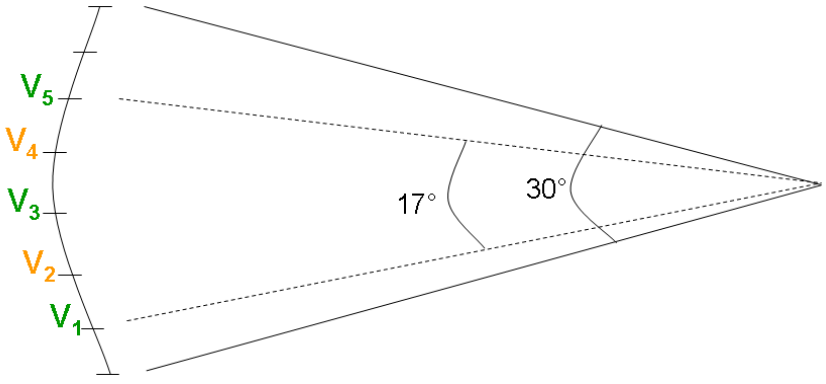


Figure 4.3: Example of input viewpoints. V_1, V_3, V_5 are the input viewpoints and V_2, V_4 are used to evaluate the quality of synthesized virtual views.

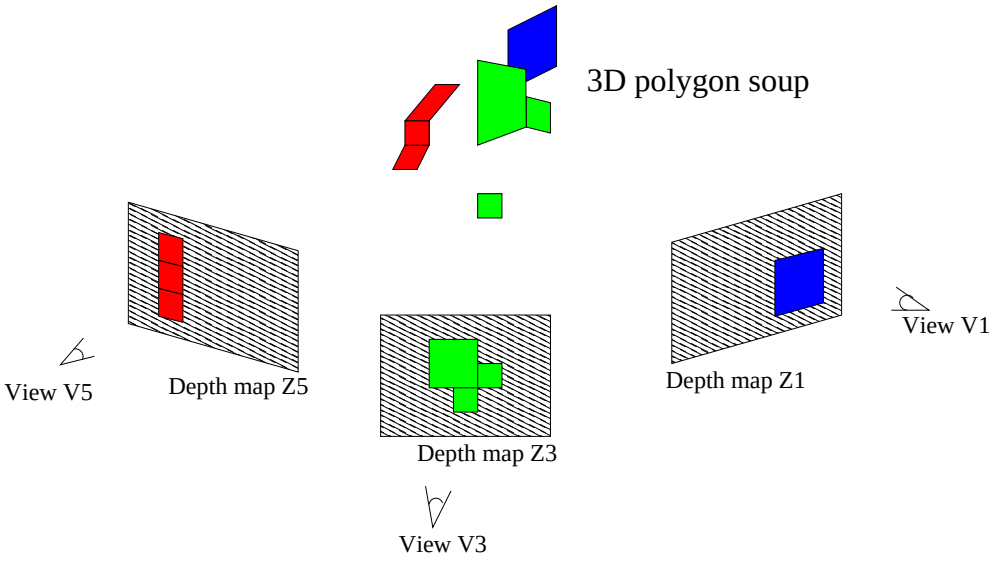


Figure 4.4: Polygon soup. Each 3D polygon is defined by a 2D polygon (quad), and by the depth information at each corner of the quad.

move redundant polygons by pruning the corresponding leaf in the tree. Moreover, the decomposition can be adapted to the geometry of the scene such that large quads approximate flat surfaces and small quads preserve geometric details and discontinuities. Finally, since quads are aligned with blocks of pixels in depth maps, it is also possible to compress the depth maps with existing block-based video compression methods, but it would involve that the polygon soup is extracted at the user side of the system.

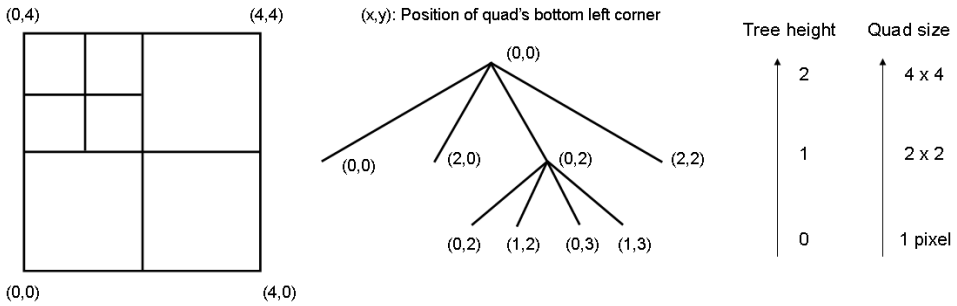


Figure 4.5: Example of image decomposition and quadtree structure. Each level of the quadtree gives the size of the quads and each node gives the position of the bottom left corner of the quads.

Texture mapping. Concerning the texture of the polygons, it is given by the original images: the block of pixels in the image corresponding to a quad is mapped onto the 3D

polygon using texture mapping techniques supported by graphics hardware. Since, multiple images are available, it is possible to texture the polygons with multiple textures depending on the view-point. This results in a multi-textured appearance of the polygon that better reproduces specular effects of real-world scenes. In addition, this kind of multi-view videos can be efficiently compressed using the standardized H.264/MVC compression method. Finally, transmitting the original views ensures maximum image quality at original view-points.

Dynamic polygon soup. We have seen in the previous chapter that both single and multiple texture/geometry representations suffer either from global reconstruction errors or inter-view inconsistencies that reduce the quality of the synthesized images. In order to deal with these drawbacks, we will present a dynamic version of the polygon soup. The idea is to allow polygons to be transformed across space and time so as to adapt to the desired view-point and be consistent with each others. Therefore, the polygon soup will be view-dependent. We call this dynamic representation 'floating polygon soup'. This dynamic representation introduces the temporal dimension, indeed polygons could be transformed not only across the views but also across time in order to adapt to the movements of the scene. An adapted compression method is then required to take into account this dynamism. It could be inspired by existing 2D video compression studies that tracks blocks of images across time and represent them using so-called 'motion tubes' representation.

4.3 Properties of the polygon soup

The proposed polygon soup can now be analyzed in terms of the general properties introduced in the previous chapter:

Construction complexity. Most of the complexity is transfered to the construction of the representation in order to achieve compactness, and to decrease the view synthesis complexity.

Compactness. Compactness is achieved by keeping the number of polygons low and reducing inter-view redundancies.

Compression compatibility. A new quadtree-based compression method will be introduced for the polygon soup. However, block-based compression is also possible providing that the quadtree decomposition step is transferred at the user side of the system.

View synthesis complexity. Virtual views are synthesized using graphics hardware and polygonal primitives. Polygons avoid the apparition of sampling artifacts compared with depth-based view synthesis, thus the view synthesis complexity is lower. Moreover, since unreliable polygons are removed during the construction of the representation, the

polygon soup reduces the ghosting artifacts and thus no additional process are required to avoid them. Additional process like hole filling and filtering are still required for high image quality.

Navigation range and image quality. Original views are preserved ensuring maximum quality at original viewpoints. Navigation in between original views is done by view-synthesis using the polygon soup as the geometry of the scene. The dynamic polygon soup adapts its geometry to every desired viewpoints in order to improve the quality of synthesized views.

4.4 Summary

In this chapter, an overview of the proposed polygon soup representation has been given. The polygon soup is a set of 3D polygons possibly disconnected or overlapping with each others. They are stored as 2D quads using a quadtree decomposition of the depth maps and redundancies across the views are adaptively reduced by pruning some leaves of the quadtree. This representation can be compressed either with a quadtree-based or block-based compression method. The view-synthesis step is performed with polygon warping and texture mapping using graphics hardware. Additional process may be needed for filling unknown areas and improving virtual views. The texture is given with the original images that are compressed with existing H.264/MVC video compression standard. Finally, a dynamic version of the polygon soup, called floating polygon soup, allows to increase the quality of synthesized images may take into account the temporal dimension of the polygons. Possible compression of this representation could be inspired from the 'motion tubes' compression method. These different processing steps are summarized in table 4.1.

In the following chapters, the details of the construction, compression and view synthesis steps will be given. The last chapter is dedicated to the description of the floating polygon soup for spatial dimension. The evaluation and compression of the temporal dimension of the representation is left as a perspective.

	Construction	**Compression**	**View-synthesis**
Polygon soup	Depth estimation; Quadtree decomposition; Redundancies reduction	Quadtree-based compression or block-based compression;	Polygon warping; Texture mapping; Post-processing; Quadtree decomposition (if block-based compression)
Floating polygon soup	Same as above + deformation estimation	Adapted compression inspired by 'motion tubes'	Same as above + floating geometry
Multiple views for texturing the polygon soup	Color correction (if needed)	Compression with H.264/MVC standard	View-dependent texture mapping on the polygon soup

Table 4.1: Summary of the processing steps with the polygon soup representation.

Chapter 5

Construction of the polygon soup

Contents

In the previous chapter, an overview of the proposed polygon soup representation was given. It is made of 3D polygons stored in 2D with depth values at each corners. Moreover, some redundant or unreliable polygons are removed from the polygon soup. This chapter explains the construction of this polygon soup from the input depth maps. It holds in two steps:

1. A quadtree decomposition of the depth maps. This step extracts a set of 2D quads for each view (section 5.1). The quadtree structure provides many advantages for compactness and compression, but special attention must be given to the decomposition method such that depth discontinuities and geometric details are preserved.

2. An inter-view redundancy reduction. This step selectively eliminates some redundant or unreliable quads (section 5.2). The aim is to increase the compactness of the polygon soup while preserving as much as possible the navigation range and image quality. To do so, some criteria are needed to select the polygons by order

of priority. This second step also helps reducing artifacts around discontinuities (ghosting artifacts).

Figure 5.1 sums up the different steps of the method. Although three views are considered here, the proposed framework applies to any number of views.

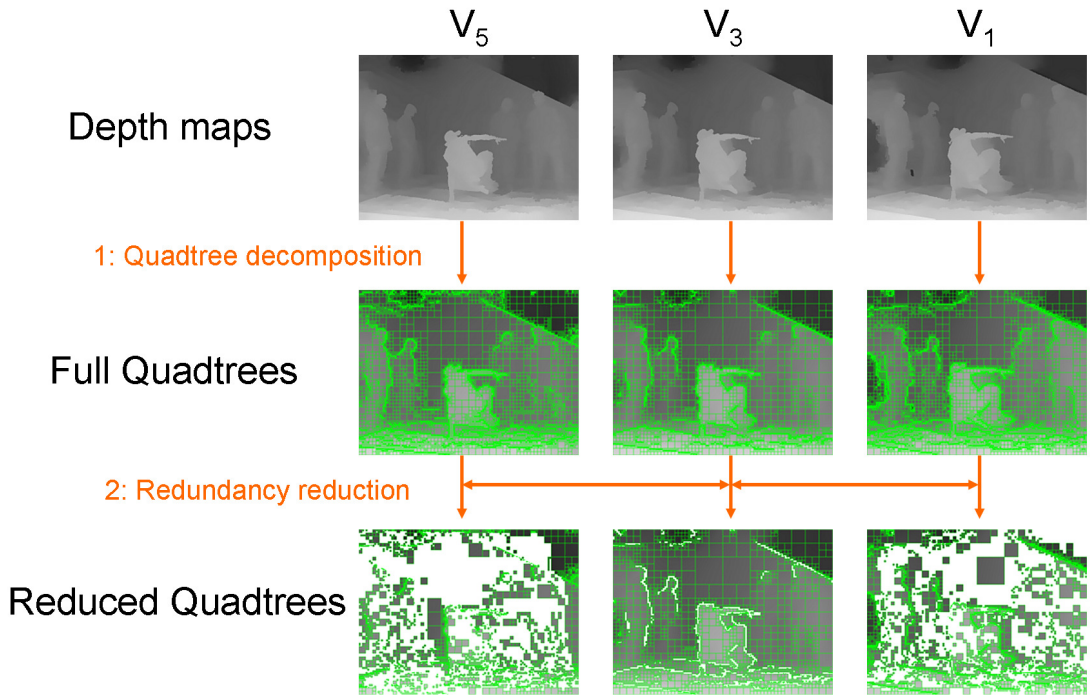


Figure 5.1: Overview of the construction method.

5.1 Quadtree decomposition

This section describes the extraction of 2D quads from the MVD data, using a quadtree decomposition.

In a quadtree, each node corresponds to a quad of a certain size and position. Then, for each quad, four corners have depth values extracted from the depth map. In our method, the quads are not necessarily connected to their neighbors because all corners have their own depth values.

The quadtree decomposition for one view aims at providing an accurate model of the associated depth map with a limited number of quads while preserving depth discontinuities. Quads are recursively scanned and divided if they contain a depth discontinuity or if they do not provide a good enough approximation of the depth map. At this step, only depth information is used. Each depth map is processed independently so that a quadtree is created for each view.

In this section, we first introduce the notion of re-projection shift as a mean to evaluate the discontinuity preservation and the geometry preservation (section 5.1.1). Then the subdivision method is detailed (section 5.1.2).

5.1.1 Re-projection shift

In order to decompose a depth map into a quadtree, some criterion is needed to evaluate the difference between the original depth map and the quadtree. Here, we introduce the notion of re-projection shift.

Definition. The 're-projection shift' is the relative shift between two pixels re-projected from one view into another one. Figure 5.2 illustrates this re-projection shift when the central view is re-projected into the left and right views. Let $p1$ and $p2$ be two pixels in one view, then their re-projection shift in view v is:

$$s(p1^v, p2^v)$$

Errors and re-projection shift. The depth maps and color images are local representations estimated for one viewpoint. Inaccuracies in this representation create artifacts as the virtual view moves aside the original viewpoint. In a word, the errors that appear in virtual views depend on the re-projection of each viewpoint. Therefore, it is important to take into account this re-projection shift during the computation of the quadtree.

5.1.2 Subdivision method

The subdivision method consists in testing if a quad should be subdivided or not, according to the criterion presented above. The subdivision method holds in two steps: discontinuity preservation and geometry preservation. During the discontinuity preservation step, the re-projection shift is used to decide whether a quad should be subdivided or not. Then, during the geometry preservation, the re-projection shift is also used for the subdivision test. These steps are summarized in figure 5.3, and detailed in the following.

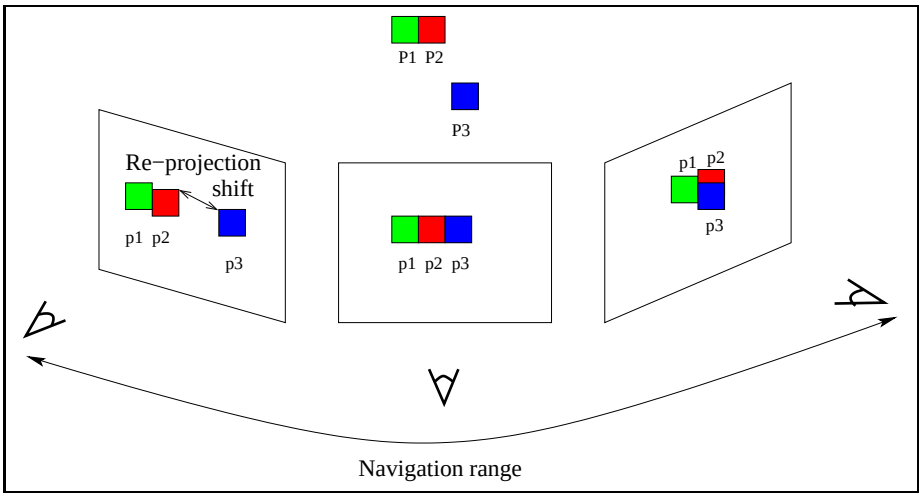
We first define some notations. Let p be a pixel in one view. Let p^l and p^r be the re-projections of p into the left-most and right-most views. $s(p1, p2)$ is the relative shift between two re-projected pixels $p1$ and $p2$.

Discontinuity preservation. Let $p1$ and $p2$ be any two adjacent pixels in quad q in one view. Considering threshold T_d, then q is subdivided if:

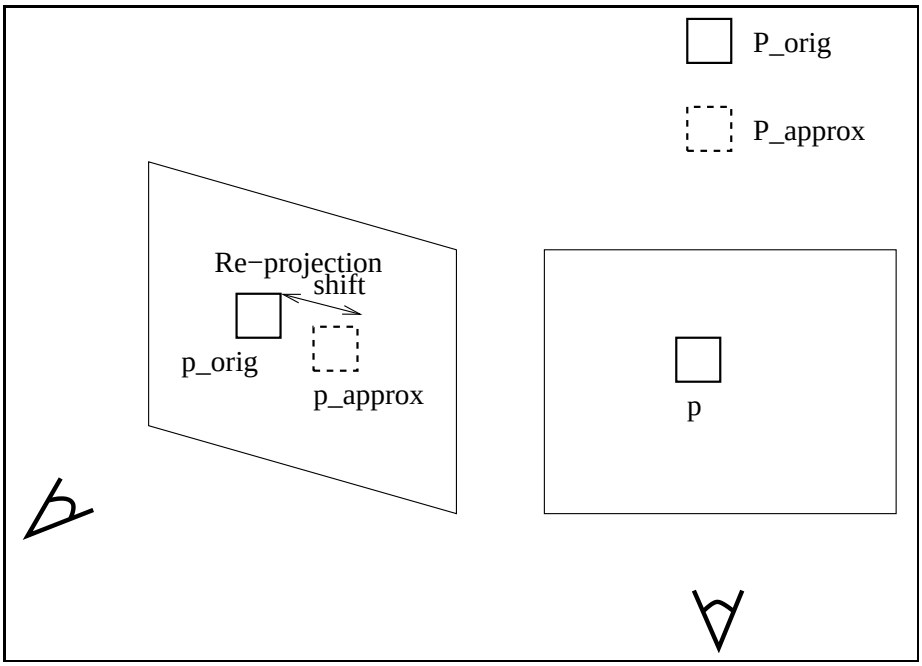
$$\max_{p1,p2} \Big(s(p1^l, p2^l), s(p1^r, p2^r) \Big) > T_d$$

where $s(p1^{l,r}, p2^{l,r})$ is the re-projection shift.

Enforcing this criterion means that the quads in the image are subdivided if neighboring pixels have large re-projection shift, and thus large depth discontinuities. Similar

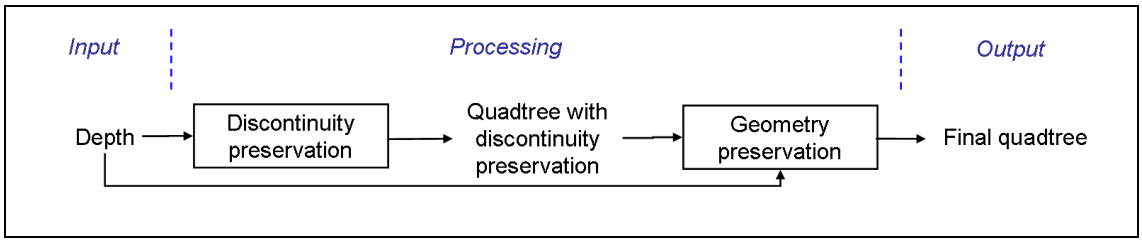


(a) Re-projection shift between neighboring pixels. The central view is re-projected into the left and right views. The re-projection shift is the relative shift between between two neighboring pixels re-projected into the synthesized view.

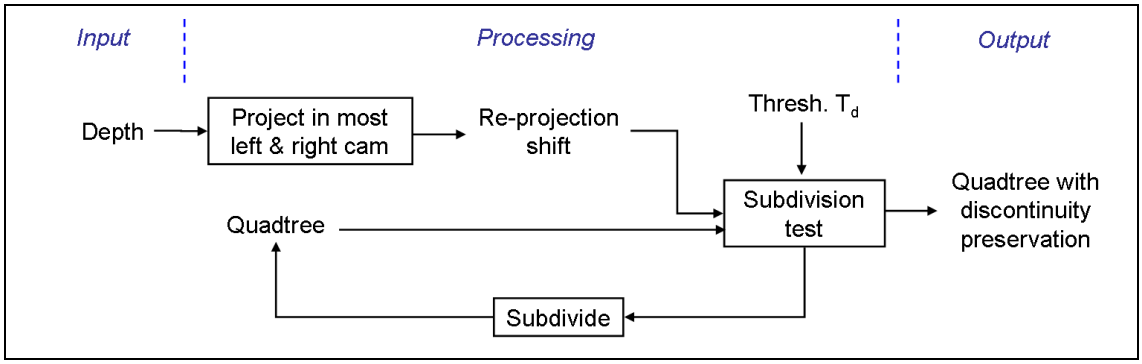


(b) Re-projection shift for one pixel with different depth values. It is the relative shift for one pixel re-projected into a synthesized view using respectively original depth and depth approximated with the quadtree.

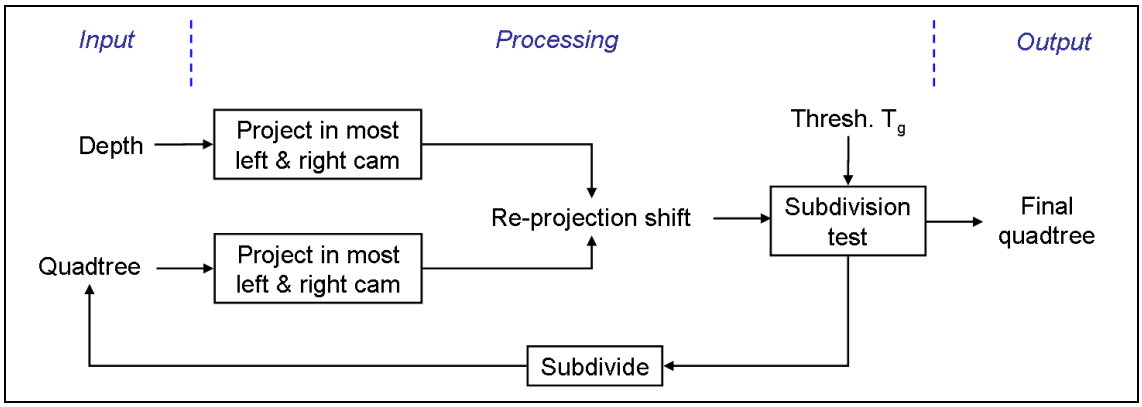
Figure 5.2: Illustration of the re-projection shift between neighboring pixels (a) and for one pixel with different depth values (b).



(a) Quadtree decomposition



(b) Discontinuity preservation



(c) Geometry preservation

Figure 5.3: Overview of the quadtree decomposition method. (a) Global method. (b) Detail of the discontinuity preservation step. (c) Detail of the geometry preservation step.

procedure is performed for each view. This criterion will lead to have small quads located close to depth discontinuities.

Geometry preservation. From the coarse representation obtained, a quad is again subdivided if the geometry approximation based on the re-projection shift is too high. First, a quad is represented as 2 triangles (the diagonal of the quad from bottom left corner to top right corner is used). The plane equations for these 2 triangles are computed using the depth values of the corners of the quad. Let p_{orig} be a pixel at position (x, y) with original depth value Z, and p_q be a pixel at the same position (x, y) but depth value Z_q on approximated by the quad. Then, we define the mean of the re-projection shift for q when projected into a view v as:

$$\epsilon^v = \underset{(p_{orig}, p_q) \in q}{Mean} \left(s(p^v_{orig}, p^v_q) \right)$$

where $s(p^v_{orig}, p^v_q)$ is the re-projection shift. The quad q is subdivided if:

$$\max(\epsilon^l, \epsilon^r) > T_g$$

with ϵ^l and ϵ^r the approximation errors after projection into the left and right views. Using this approximation criteria, local errors inside a quad are accepted as soon as the mean error stays low.

Quad size limitation. In order to limit polygonal artifacts and local geometry errors inside a quad, a parameter $QSmax$ that sets the maximum size of the quads is introduced. Therefore, any quad will have a size S_q smaller or equal to $QSmax$:

$$S_q \leq QSmax$$

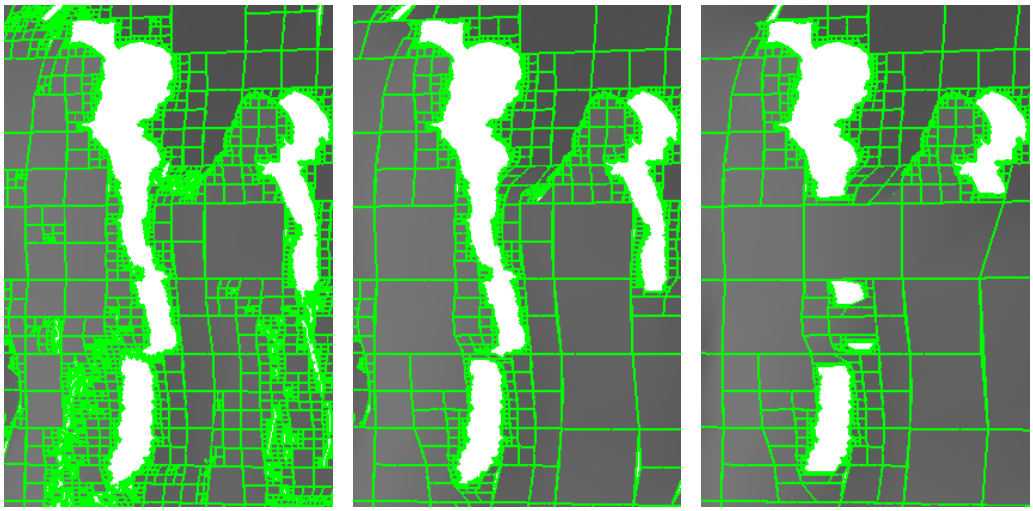
Note that both "geometry preservation" and "quad size limitation" deal with the compromise between the geometry approximation and the number of quads: when quads are bigger then there are less quads and more geometry errors.

5.1.3 Results

This section now gives the results of the quadtree decomposition and illustrates the compromises involved by the thresholds T_d, T_g and $QSmax$.

Discontinuity preservation. The threshold T_d controls the compromise between the discontinuity preservation and number of quads. This compromise is now illustrated. First, the geometry preservation step is disabled and the constraint on the maximum size of the quads is also disabled. Figure 5.4 shows a detail of the quadtree decomposition of view V_5 after re-projection into V_1 for *Breakdancers* sequence. Three different values of T_d have been tested. This figure illustrates that small discontinuities are unnecessary preserved if T_d is low (figure 5.4(a)) which leads to a high number of quads, and that large discontinuities are not preserved if T_d is high (figure 5.4(c)). In

the remaining of this thesis, the value of T_d is empirically chosen to $T_d = 10$ pixels (figure 5.4(b)), however this parameter may be adjusted according to the desired bitrate after the compression stage.

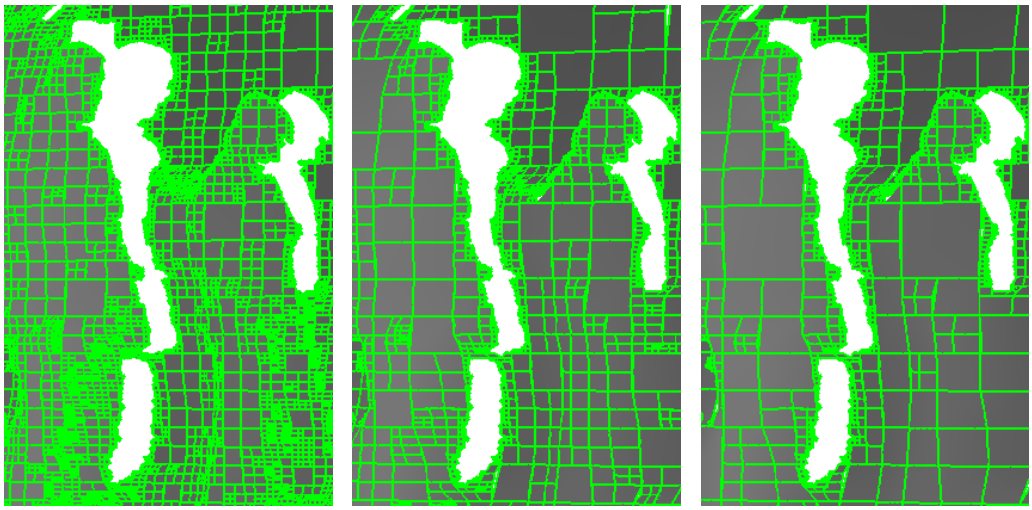


(a) $T_d = 4$ pix, 46701 quads (b) $T_d = 10$ pix, 19089 quads (c) $T_d = 25$ pix, 14133 quads

Figure 5.4: The value of T_d controls the compromise between the discontinuity preservation and the number of quads. When T_d is low, small discontinuities are preserved which leads to many quads subdivisions (a). When T_d is high, depth discontinuities are not preserved, thus the quads connect the foreground to the background (c).

Geometry preservation. The threshold T_g controls the compromise between the geometry preservation and the number of quads. This compromise is now illustrated. At this stage, the discontinuity preservation has been processed and the constraint on the maximum size of the quads is disabled. Figure 5.5 shows a detail of the quadtree decomposition of view V_5 after re-projection into V_1. Three different values of T_g have been tested. This figure illustrates that the geometry is preserved if T_g is low (figure 5.5(a)) but the number of quads is high. On the other hand, the geometry is not preserved if T_g is high (figure 5.5(c)) but the number of quads is low. In the remaining of this thesis, the value of T_g is empirically chosen to $T_g = 1$ pixel (figure 5.5(b)), however this parameter may be adjusted according to the desired bitrate after the compression stage.

Quad size limitation. The maximum quad size $QSmax$ controls the compromise between resolution of the quadtree and the number of quads. This compromise is now illustrated. At this stage, the discontinuity preservation and geometry preservation have been processed. Three different values of $QSmax$ have been tested. Figure 5.6 illustrates that the resolution of the quadtree is high if $QSmax$ is low but the number of quads is high (figure 5.6(a)). On the other hand, the resolution is not low if $QSmax$



(a) $T_g = 0.5$ pix, 44022 quads (b) $T_g = 1$ pix, 24141 quads (c) $T_g = 2$ pix, 19533 quads

Figure 5.5: The value of T_g controls the compromise between the geometry preservation and the number of quads. When T_g is low, the geometry is preserved but the number of quads is high (a). When T_g is high, the geometry is not preserved, and the number of quads is low (c).

is high but the number of polygons is low (figure 5.6(c)). In the chapter 7, the value of the parameter $QSmax$ will be adjusted to the desired bitrate.

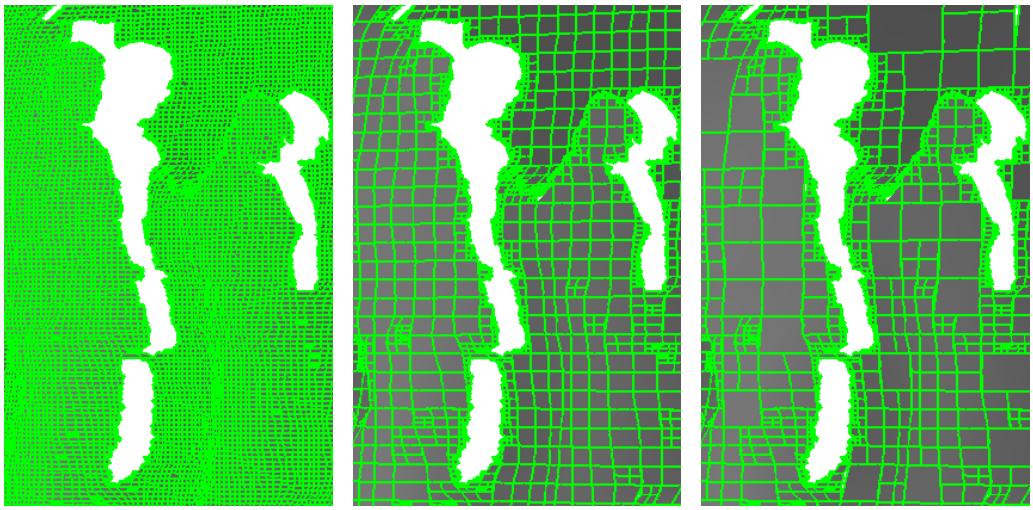
Final quadtree decomposition. After applying the discontinuity, geometry and smoothness preservation steps, one quadtree decomposition is obtained per input views. Figure 5.7 shows the final quadtree decomposition obtained with the parameters $T_g = 1$ pixel, $QSmax = 16$ pixels and $T_d = 10$ pixels for *Breakdancers* and *Ballet* and $T_d = 4$ pixels for *Book Arrival*.

These thresholds have been tuned empirically: an appropriate compromise between the number of quads and re-projection shift has been chosen and validated with the subjective evaluation of the quality of synthesized views.

The number of quads obtained with these parameters and for one frame of the sequences are given in table 5.1. The number of quads used for the central view is much lower than for the lateral views. This is an advantage of taking into account the re-projection shift for the quadtree decomposition: for the central view, the re-projection shift is smaller than the lateral views.

5.1.4 Summary and discussions.

This section has presented our proposed method for decomposing the input depth maps into quadtrees. Has already explained, the quadtree structure offers several advantages for compactness and compression. However, the quads must preserve the depth discontinuities and geometric details of the scene. For this purpose, a criterion based on the



(a) $QSmax$ = 4 pix, 64371 quads (b) $QSmax$ = 16 pix, 25260 quads (c) $QSmax$ = 64 pix, 24141 quads

Figure 5.6: The value of $QSmax$ controls the compromise between the resolution of the quadtree and the number of quads. When $QSmax$ is low, the resolution is high but the number of quads is high (a). When $QSmax$ is high, the resolution is not low, and the number of quads is low (c).

	Breakdancers	*Ballet*	*Book Arrival*
Right view	26505	29091	29082
Central view	18792	20262	19218
Left view	25260	27288	27834
Total	70557	76641	76134

Table 5.1: Number of quads after quadtree decomposition for one frame of sequences *Breakdancers* and *Ballet* and *Book Arrival*.

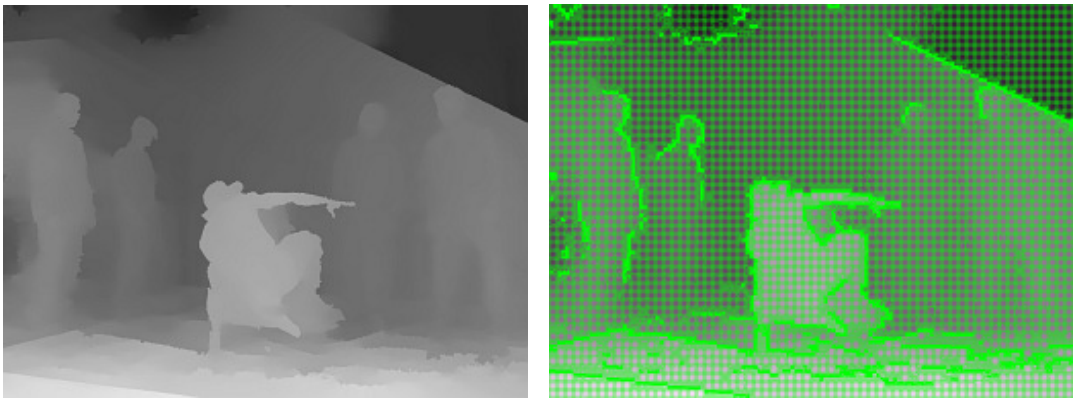


Figure 5.7: Original depth map and quadtree decomposition of *Breakdancers*.

re-projection shift has been proposed. Using this criterion, the quadtree decomposition respects three criteria associated with three thresholds: discontinuity preservation, geometry preservation and quad size limitation. The first criterion prevents the connection of foreground and background objects and therefore avoids strong deformations in synthesized views. The two other criteria control the quality of the geometry approximation.

On the contrary to classical criteria based on the depth error, this criterion takes into account the re-projection shift of each view within the navigation range. It results in an equivalent quality of approximation of the scene within the navigation range, but with different quadtree decomposition: more quads for lateral views than the central one. Therefore, the trade-off between the number of quads and the quality of geometry approximation is adapted to each view, according to their positions (wide-baseline or small-baseline), and without changing the threshold values for each views.

With the proposed method, the choice for the threshold values is done empirically by finding a compromise between the number of quads and the re-projection shift which is validated by subjective evaluation of the quality of synthesized views. The evaluation of synthesized views is very important because the measures of geometry errors do not necessarily match with the quality of synthesized images. Therefore, a more suitable error measure for the quadtree decomposition could be based on the quality of synthesized views compared with original ones. In this case the procedure would be quite similar to the proposed one, i.e. the error would focus on re-projected data so that the navigation range can be considered. However, the comparison would be performed between the synthesized views and the original ones rather than between projected quads and projected depth maps. Nevertheless, one particular issue has to be considered with such a method: if an error measure such as PSNR is used, then even small geometry errors (that result in small pixel shift after re-projection compared to original) can give high error values in textured areas. This error would not be representative of the geometry distortion, resulting in a lot more quads than necessary. Therefore, special attention has to be given to the choice of the error measure so that

it is adapted to the properties of synthesized views.

Finally, for an automatic tuning of the threshold values, an optimization method that finds the best compromise between the number of quads and the error measure could be used.

5.2 Redundancy reduction

The quadtree decomposition obtained previously gives a redundant polygon soup, i.e. a set of quads with inter-view redundancies, which is not compact. Here, the term *inter-view redundancy* means that some quads defined in different views represent the same scene content, thus they are redundant. In other words, a quad is not redundant if at least a part of its surface is not already represented by the other polygons. Thus if two polygons partially overlap, they can have both redundant and non-redundant areas. The goal of this section is now to explain how the redundancies are reduced.

The main idea is to iteratively generate a polygon soup by selecting quads in a certain priority order. One iteration of the selection is illustrated in figure 5.8: first, an original view is predicted using the current polygon soup and view synthesis method. This prediction shows that some information is missing in the polygon soup compared with the original view. Then, in order to fill this missing information, the corresponding quads extracted previously from the original view are selected. As a result, the new polygon soup is more complete. This process is then iterated with another view.

In addition to the reduction of the redundancies, this process also enables to eliminate unreliable quads that create artifacts at the view synthesis stage. For example, ghosting artifacts around object boundaries are eliminated. These results will be detailed in the view synthesis chapter (chapter 6).

In the following of the section, the properties and method of the redundancy reduction will be explained. Then the results of this process will be evaluated in terms of the number of quads.

5.2.1 Priority order for the quads

To better understand the redundancy reduction process, three criteria are defined for ordering quads by priority.

The view order is important. The original views define which part of the scene will be described. The central view describes most area of the scene, except areas that are occluded by foreground objects. On the other hand, the lateral views contain a portion of these occluded areas. During the redundancy reduction process, the polygon soup is initialized with the central view, which we call the reference view, and the lateral views are used to fill the missing areas. Since the representation is built incrementally, then any view that has already been processed contributes to the representation before the next view is processed. Therefore the furthest view from the reference one should be processed before the closest. This scanning order aims to reduce the overall number of quads: for instance an area which is progressively disoccluded will be represented by

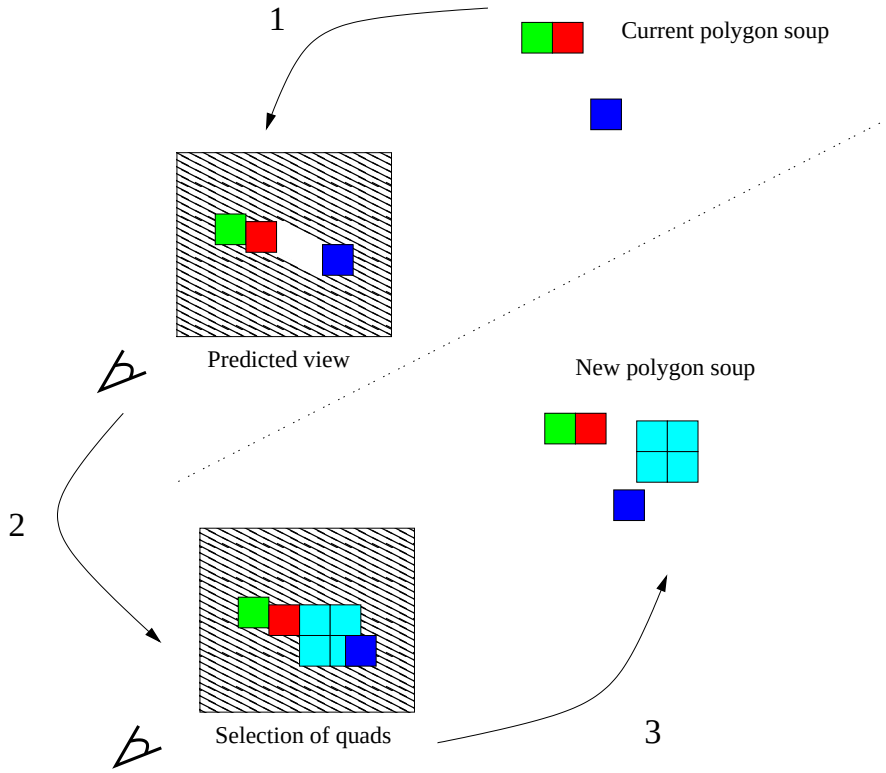


Figure 5.8: One iteration of the quads selection. 1- Prediction of original view using the current polygon soup. The white area shows the missing information (e.g disocclusion area). 2- Some quads extracted from the original view are selected to fill the missing information. 3- As a result, the new polygon soup is more complete.

one large quad from the furthest view, rather than a succession of smaller quads, one from each closer view.

Small quads are unreliable. Typically, quads situated around depth discontinuities are small and unreliable because they contain mixed color between foreground and background objects. This color mix is often responsible for the ghosting artifacts in synthesized views. Moreover, small quads often correspond to 3D surfaces whose normal vector is parallel to the image plane. Such 3D surfaces are more reliably represented using lateral view points (see figure 5.9). In order to preferably select quads which

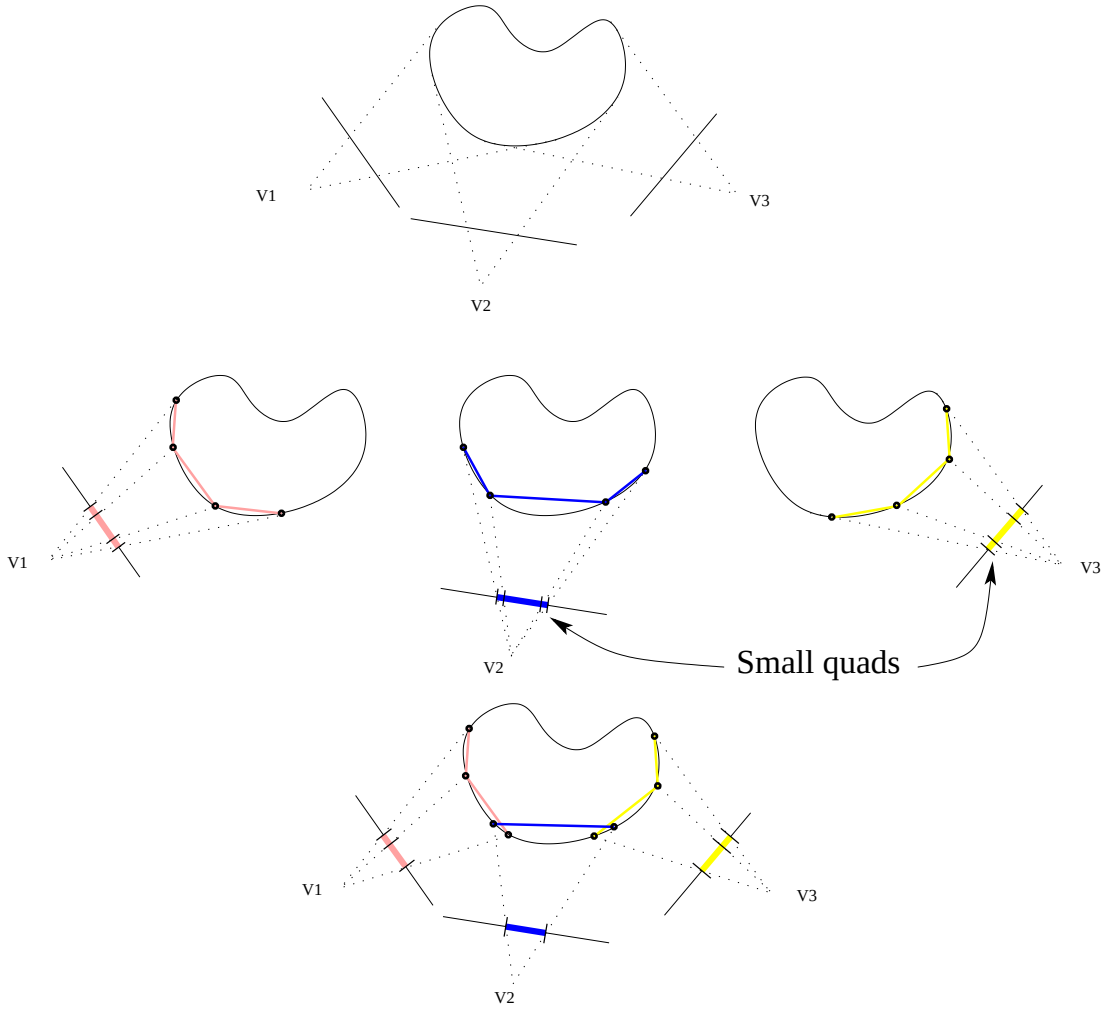


Figure 5.9: Unreliable quads : considering 3 viewpoints (top), small quads are situated near depth boundaries in each view (middle), redundancy reduction selects preferably large quads and represent the 3D surface with good resolution. Small quads are used only if necessary (bottom).

are both large (to reduce coding cost) and reliable (to enhance rendering quality), the selection operates as follows: first, all the small quads from the reference view are discarded because this part of the scene may be better represented from the side views. Second, quads from the side views are iteratively selected depending on their size: bigger quads from side views are added before the smaller quads. Finally, if some information is still missing in the reference view, then the small quads previously discarded are selected again and added to the polygon soup.

Disocclusions first, then foreground. At each iteration, a new view is used to fill the missing information. This missing information may be of two types: either a disocclusion area (i.e. an empty area) or a foreground area (i.e. a foreground area is visible in the view, but a background area is predicted with the current polygon soup). In order to preferably select large and reliable quads, disocclusion areas should always be filled before the foreground is added. Therefore, the redundancy reduction operates in two rounds: first the disocclusion areas from all the views are filled in a first round, and then the foreground areas are added only in a second round.

To summarize, the redundancy reduction process is initialized with a reference camera, and the other views are processed from the furthest to the closest to the reference. Only useful quads are selected in a decreasing order of their size and in the processing order of the views. Two rounds of selection are processed: first, the quads are selected to fill the disocclusion areas, second, the quads are selected to add the missing foreground information.

This process is close to the one presented in [MSD+08] for building the LDV representation. However, the first difference is that the discarded quads are not only in the background but also in the foreground regions. The second difference is that the representation is built incrementally. Finally, the last difference is that the data is not projected and stored into the reference frame: the texture of the quad is represented as a square block in its original view, thus avoiding texture degradation due to resampling occurring with such projection.

As this redundancy reduction process is initialized with a reference view and the lateral views are used mostly for disocclusion areas, then the range of navigation is limited around the reference view. Thus, we make the assumption that the lateral views are situated in that range of navigation. For example if full navigation is desired all around a scene, then using only one reference view is not enough and multiple reference views would have to be defined together with their lateral views. Therefore, these hypothesis are related to the application context of this thesis.

5.2.2 Reduction method

The three rounds of selection used for redundancy reduction are now detailed. Namely, the initialization, the filling of the disocclusions and addition of the foreground.

Initialization Let Ω_k be the polygon soup at iteration k, and q a quad from the reference view V_{ref} (generally the central view). V_{ref} is used to define the initial

polygon soup Ω_0. As explained above, small quads are unreliable, thus they are not selected during this initialization. This can be formulated as:

$$\Omega_0 = \{q \in V_{ref}, S_q \geq QSmin\}$$

where S_q is the size of q, and where $QSmin$ is the threshold for minimum size of the quads.

Filling the disocclusions Ω is iteratively completed by selecting quads from V_i that fill disocclusion areas. Let k be the current iteration, and Ω_{k-1} the polygon soup from previous iterations (figure 5.10(left)). Ω_{k-1} is first projected onto view V_i (figure 5.10(center)). The resulting image contains disocclusion areas, i.e. empty areas in white in the figure. Then, quads from V_i that help filling the disocclusion areas are added to Ω_{k-1} (figure 5.10(right)). This can be formulated in the following equation:

$$\Omega_k = \Omega_{k-1} \cup \{q \in V_i, S_q \geq QSmin, \exists p \in q | p \notin \mathcal{P}_i(\Omega_{k-1})\}$$

where $\mathcal{P}_i(\Omega_{k-1})$ denotes the projection of the current set of selected quads Ω_{k-1} onto view V_i. The test $p \notin \mathcal{P}_i(\Omega_{k-1})$ detects if a pixel p overlaps a disocclusion area.

This round of iterations is repeated for decreasing size of the quads $S \in [QSmin, QSmax]$. Since view order is important, the last processed view is the reference one (all side views have been processed before). This way, small quads unselected during the initialization are added only if necessary. This method is summarized in algorithm 1.

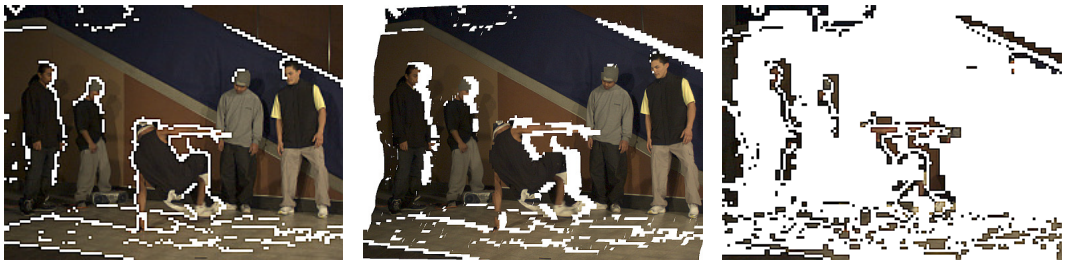


Figure 5.10: One iteration of filling disocclusions. Left: Ω_{k-1} polygon soup at iteration $k-1$. Center: Ω_{k-1} projected in V_i (Disocclusions). Right: Selected quads from V_i.

Adding the foreground The process of adding the missing foreground areas is similar to filling the disocclusion areas: side views are processed and quads are added in decreasing order of their size. Then the reference view is processed. In this case, a depth test is performed instead of disocclusion test to detect if a quad belongs to a foreground area or not:

$$\Omega_k = \Omega_{k-1} \cup \{q \in V_i, S_q \geq QSmin, \exists p \in q | Z(p) < Z(\mathcal{P}_i(\Omega_{k-1}))\}$$

where $Z(p) < Z(\mathcal{P}_i(\Omega_{k-1}))$ detects if pixel p has smaller depth than the current depth obtain by projection of Ω_{k-1} in V_i. If yes, then q belongs to the foreground.

Figure 5.11 shows the different steps of the process on a detail area of the *Breakdancers* sequence.

Objective: Parse quads in a certain order and fill disocclusions.

input : Initial polygon soup Ω_0. Initial threshold $QSmin$

```
1  k = 1;
2  while QSmin >= 1 do
3  |  foreach view V_i (only side views) do
4  |  |  Project Ω_{k-1} in V_i ;
5  |  |  foreach quad q ∈ V_i, S_q ≥ QSmin do
6  |  |  |  if fillDisocclusion (q) then
7  |  |  |  |  Select q and add in Ω_k ;
   |  |  |  end
   |  |  end
8  |  |  k = k + 1;
   |  end
9  |  QSmin = QSmin/2;
   end
   // In the end, process the reference view
10 Project Ω_{k-1} in V_ref ;
11 foreach quad q ∈ V_ref do
12 |  if fillDisocclusion (q) then
13 |  |  Select q and add in Ω_k ;
   |  end
   end
```

Algorithm 1: Filling the disocclusions

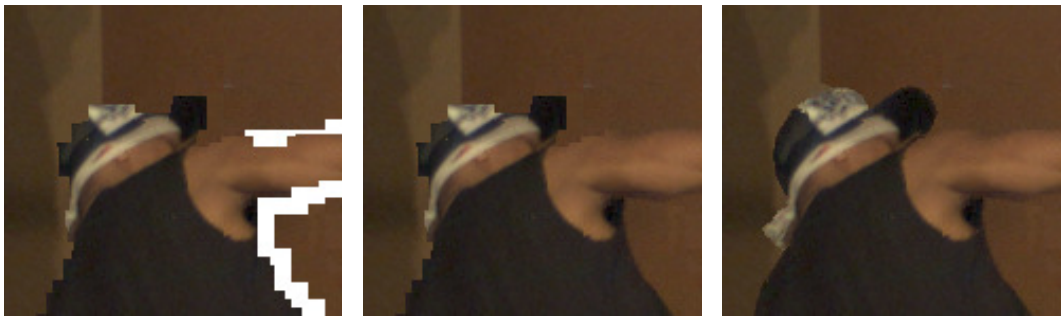


Figure 5.11: The representation is completed step-by-step. (left) Both disocclusion and foreground information are missing. (center) Disocclusions have been filled. (right) Foreground has been added.

5.2.3 Results

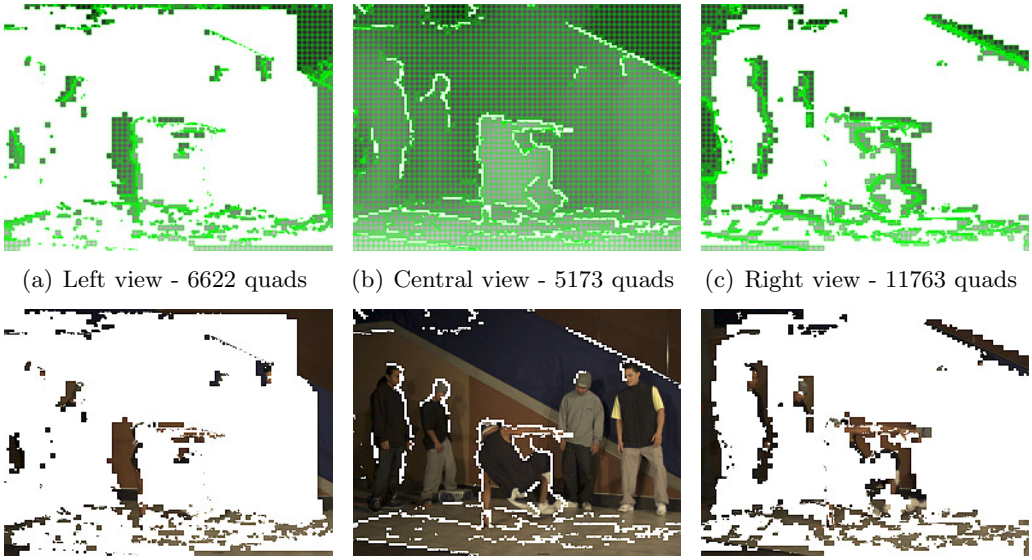
The goal of this section is to evaluate the redundancy reduction by analysing the number of quads before and after the reduction. The results of image quality after view synthesis will be given in chapter 6.

The tests of the redundancy reduction were performed on sequences *Breakdancers*, *Ballet* and *Book Arrival*, with a configuration of 3 views where the central one is considered as the reference. In the redundancy reduction process, the threshold for the minimum size of the quads was initialized to $QSmin = 8$.

Figures 5.12, 5.13 and 5.14 show the final polygon soup, i.e. the final set of selected quads in each view. As expected, a lot of quads have been removed. A bigger area is covered in the reference view, and quads from the side views provide information on disoccluded areas.

The redundancy reduction has efficiently reduced the number of quads. Table 5.2 gives the number of quads for each view after the redundancy reduction for one frame of the sequences. The central view has the lowest number of quads because most of the small quads in this view have been removed, and mainly big quads away from depth discontinuities are present in this view.

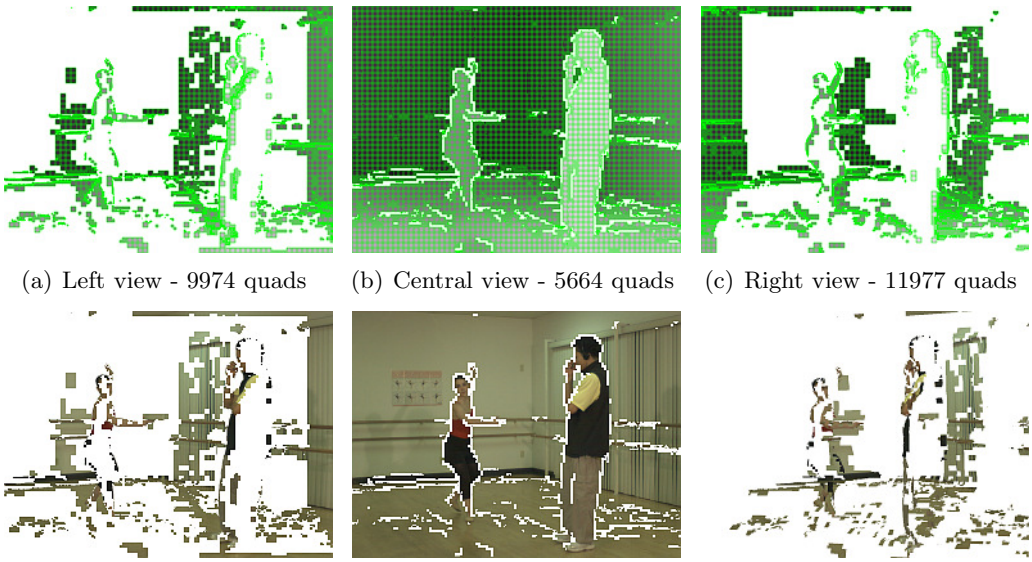
Table 5.3 gives the number of quads before and after the redundancy reduction, as well as the average number of quads for one view before the redundancy reduction. About 65% of the quads have been removed compared with the full 3 views. Moreover, compared to a single view, the increase is only of 8% at most.



(a) Left view - 6622 quads (b) Central view - 5173 quads (c) Right view - 11763 quads

(d) Left view (e) Central view (f) Right view

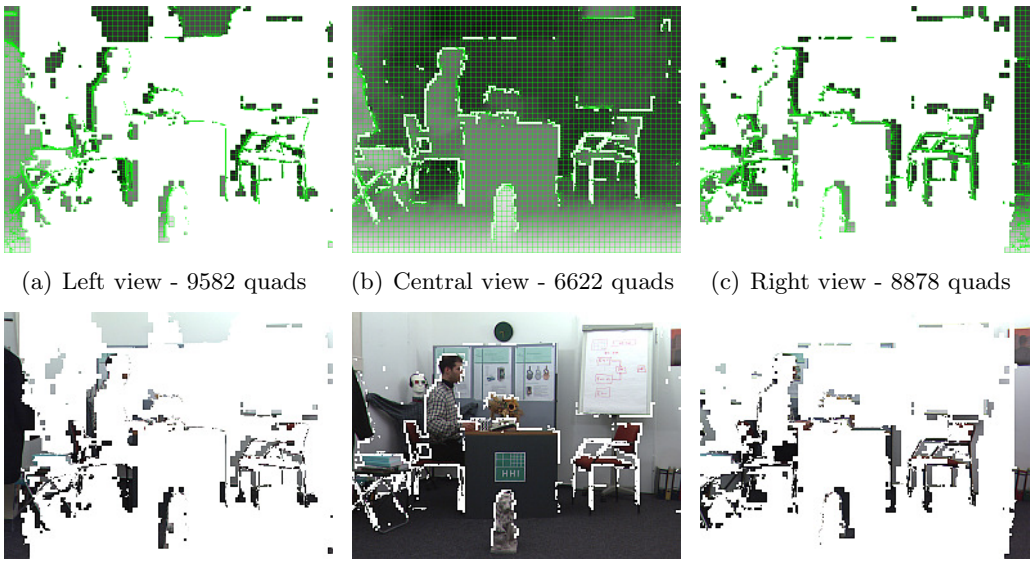
Figure 5.12: Result of quads selection for *Breakdancers*: final set of selected quads in each view (a) (b) (c) and associated color information (d) (e) (f).



(a) Left view - 9974 quads (b) Central view - 5664 quads (c) Right view - 11977 quads

(d) Left view (e) Central view (f) Right view

Figure 5.13: Result of quads selection for *Ballet*: final set of selected quads in each view (a) (b) (c) and associated color information (d) (e) (f).



(a) Left view - 9582 quads (b) Central view - 6622 quads (c) Right view - 8878 quads

(d) Left view (e) Central view (f) Right view

Figure 5.14: Result of quads selection for *Book Arrival*: final set of selected quads in each view (a) (b) (c) and associated color information (d) (e) (f).

	Breakdancers	*Ballet*	*Book Arrival*
Right view	11763	11977	8878
Central view	5173	5664	6622
Left view	6622	9974	9582
Total	23558	27615	25082

Table 5.2: Number of quads after redundancy reduction for one frame of sequences *Breakdancers*, *Ballet* and *Book Arrival*.

	BreakDancer	*Ballet*	*Book Arrival*	
Full views	70557	76641	76134	↓ from -64 to -67%
Reduced views	23558	27615	25082	↑ from 0 to +8%
1 view	23519	25547	25378	

Table 5.3: Number of quads before and after the reduction.

5.2.4 Summary and discussions

In this section, the proposed method reduces inter-view redundancies such that the selected quads provide a unique information at least from a small area of its surface. In the polygon soup, the non-redundant areas of the polygons can overlap with other polygons extracted from different views. Thus the polygon soup is more compact while ensuring smooth combination of all the overlapping parts.

For the redundancy reduction, a priority order has been defined depending on the view position, the quad's size, and depth of the quads. This priority order aims at selecting the more reliable quads and as few quads as possible. A particular attention has been focused on removing small quads around depth discontinuities that often contain mixed foreground/background colors and are less reliable.

Results have shown that redundancies have been efficiently reduced since about 65% of quads have been removed compared to the full polygon soup. Despite of this reduction, the original viewpoints can be reconstructed without holes using the polygon soup which proves that relevant data has not been supressed.

In this section, the selection of quads using an error measure has not been presented. Despite the priority order of this method, geometry errors due to inaccuracies of the depth maps or quad approximation may result in texture misalignments or deformation in the synthesized views. Therefore, using an image-based error measure of synthesized views could help defining reliable quads and tuning the trade-off between compactness and image quality. As already discussed in the previous section 5.1.4, special attention has to be given to the choice of the error measure so that it is adapted to the properties of synthesized views.

5.3 Conclusion

This chapter has presented the construction of the polygon soup. First, a set of quads is extracted using a quadtree decomposition of the depth maps. This step preserves depth discontinuities and geometric details while keeping as few quads as possible. To do so, an error measure based on the re-projection shift has been presented. Results have shown that it allows to better control the compromise between compactness and geometry approximation. Second, the reduction of inter-view redundancies has been presented. It uses a priority order to select reliable quads while keeping as few quads as possible. Results have shown that about 65% of quads have been removed compared to the full polygon soup.

A possible improvement of the construction would be to use an image-based error measure so that the quality of synthesized views is taken into account for tuning the parameters. However, to prevent excessive subdivisions of the quads, such a measure should be adapted to the properties of synthesized views and to the relations between texture and geometry errors.

In the following, the evaluation of the quality of synthesized views using such polygon soup will be given in the next chapter dedicated to the view synthesis step. Then the advantages provided by the compactness will be evaluated in the chapter 7 that

introduces an adapted compression method.

Chapter 6

Virtual view synthesis

Contents

In the previous chapters, an overview of the polygon soup as well as its construction method have been presented. We have seen that the polygons are extracted and selected with compactness and geometry preservation in mind. In this chapter, a view synthesis method adapted to this polygon soup is detailed and the quality of the synthesized views is evaluated. The method takes as an input the polygon soup and the original color images.

As seen in the section 2.4, synthesizing virtual views raises many issues: usually the data has to be projected into the correct viewpoint and multiple contributions of the same area have to be correctly combined. Also typical artifacts require dedicated processing and thus increase the view synthesis complexity. This chapter deals with these issues. It is divided into four sections:

- The first section is dedicated to the projection method using a single view and its associated polygons. The differences between depth-based and polygon-based projection would be highlighted. In addition, a subdivision of the quads would be detailed in order to avoid cracks between polygons of different size.

- The second section deals with the use of multiple views. Many polygons extracted from multiple views partially overlap with each others when projected onto a virtual view. A method that blends these overlapping polygons is presented. Moreover, a typical artifact that usually appears around depth discontinuities when blending multiple views is the so-called ghosting artifact (or corona artifact). The polygon soup has been designed to avoid these artifacts. This section will evaluate the effectiveness of this solution.

- The third section concerns the enhancement of the virtual views by image processing methods. Synthesized views often contain remaining disoccluded areas (unknown areas) that couldn't be predicted by any of the original views. A typical hole filling method called 'inpainting' is applied to fill these areas. Finally, an additional filtering process is performed to provide a more natural appearance around object boundaries.

- To finish, the fourth section evaluates the quality of the synthesized images. A subjective description of the image quality and artifacts is given, followed by an objective quality measure widely used in the field of video coding.

The three main processing steps of this chapter are summarized in figure 6.1. Most of the techniques presented in this chapter are known and already used for other applications or other representations. Therefore, the main contributions of this chapter is the adaptation of these techniques to the polygon soup representation, and the evaluation of the quality of virtual views synthesized using this representation.

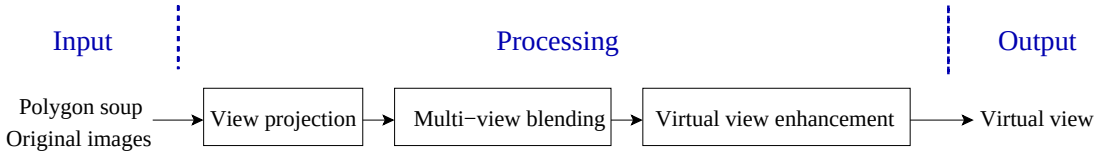


Figure 6.1: Overview of the view synthesis stage.

6.1 View projection

This section describes the projection principles used for synthesizing a virtual view. Only one view made of a color image and its associated polygon soup are considered here. The type of geometric primitive used for the projection influences the resulting artifacts that appear in the virtual view. Therefore, a comparison of the depth-based projection and polygon-based projection is done. Finally, a subdivision method that avoids cracks between quads of different size is detailed. Figure 6.2 gives an overview of the view projection step.

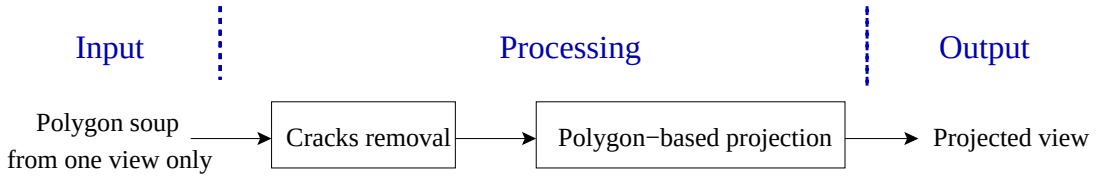


Figure 6.2: Overview of the view projection stage.

6.1.1 Projection principles

Projection from 3D to 2D. In this paragraph, we briefly explain general projection principles. Figure 6.3 illustrates the acquisition of a 3D point by a camera represented with its optical center and image plane. When a 3D point in the scene is captured by a camera, then a 2D point (pixel) is obtained in the image plane of the camera. This process is called *projection* and can be mathematically modeled using the camera position and parameters and the 3D position of the point. The multiple views of the video are the results of this projection of the 3D scene into each camera. Therefore, if the 3D geometry of the scene is known, then any view can be synthesized by projection of the 3D points into the desired view.

During the view synthesis process, multiple 3D points may be projected into the same pixel position, e.g. when a foreground object occludes the background from the desired view. Special care has to be taken so that only the front pixel with smallest depth is displayed in the view. Using a Z-buffer or an occlusion compatible scanning order are two possible solutions for this issue [Mor09].

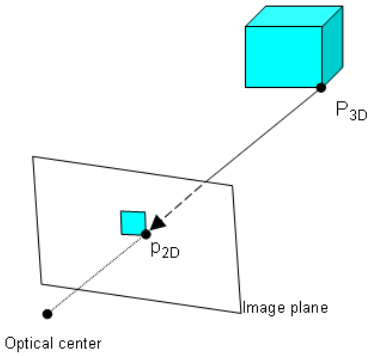


Figure 6.3: Projection of a 3D point into the image plane of the camera.

Depth-based view projection. When synthesizing virtual views using depth maps, each pixel of the original views has to be reconstructed into 3D before being projected into the desired view. We call this process back-projection since it is the inverse of the projection. It is performed using the camera position and parameters as well as the pixel position and depth. As a result, each pixel is re-projected from one view to

another one. During this process, a unique RGB color is associated to each pixel.

Polygon-based view projection. The polygon soup presented in this work is a set of quads defined with 2D plus depth data. Similarly to the depth-based view synthesis, each corner of the quads has first to be back-projected into 3D and then projected into the desired view. However, at this stage, only corners have been projected, therefore a last step is needed to convert the projected quads into a pixel image. This is called rasterisation. In addition, each corner is associated with a texture coordinate in the image such that the texture is mapped onto the quad in the synthesized view. Rasterisation and texture mapping are implemented with openGL and run on a graphics processing unit. The term 'rendering' is generally used in the computer graphics community to describe this generation of an image from a 3D model.

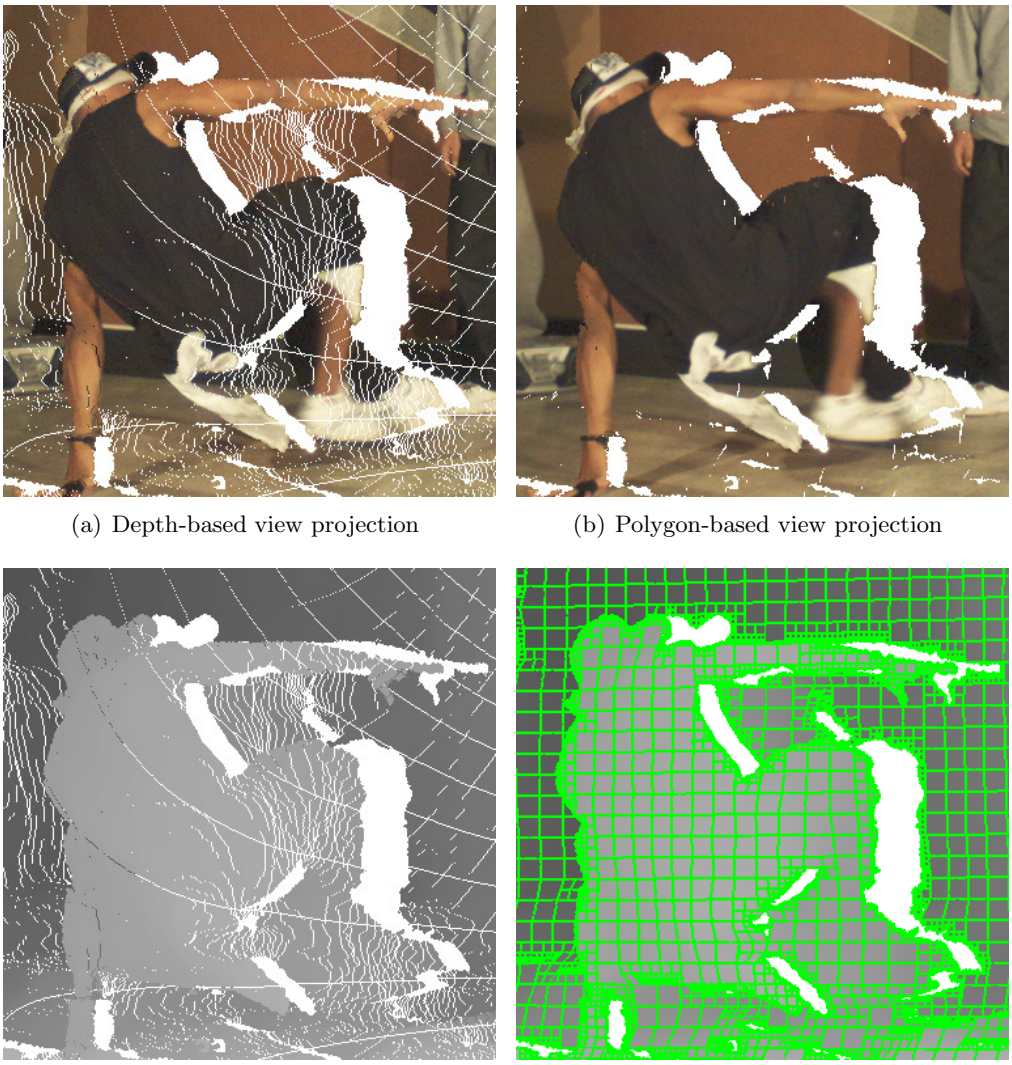
6.1.2 Depth-based vs polygon-based view projection

Figure 6.4 illustrates the difference between point-based and polygon-based view synthesis when only one original view is used. Here, the central view V_3 is re-projected into the right view V_1 using a depth map (a) and the quadtree (b). For illustration purpose, here the quadtree that is used is the one obtained before the redundancy reduction method. In both sub-figures (a) and (b), large white areas correspond to disoccluded areas, i.e. unknown areas that where occluded in view V_3. These disoccluded areas could be filled using the information coming from another view (see next section 6.2). However, with the depth-based view projection (a), many white lines appear all across the image. They are due to the sampling of pixel positions and depth values onto integer values. We call these lines *sampling artifacts*. On the contrary, with the polygon-based view projection (b), no sampling artifact is visible but still few artifacts appear in the form of small and rather vertical *cracks*. These cracks are vertical because of the horizontal displacement of the virtual camera. If the displacement was vertical, then horizontal cracks would appear. The elimination of these cracks would be explained in the next subsection.

The comparison between depth-based and polygon-based view projection shows that the polygons better model the continuity of the surface of the scene. Indeed, no sampling artifacts appear in the re-projected view. This advantage helps to reduce post-processing steps that would be necessary to eliminate sampling artifacts. Thus it reduces the view synthesis complexity. Typical elimination of the sampling artifacts consists in a depth test and median filtering process [SMD+08].

6.1.3 Elimination of cracks

After the polygon-based view projection process, cracks appear in the form of thin white lines or white triangles. They appear when two neighboring quads have different sizes because the smaller quads are not connected to the center of the bigger one, as illustrated in figure 6.5 and 6.6.



(a) Depth-based view projection

(b) Polygon-based view projection

Figure 6.4: Depth-based and polygon-based view projection. The view V_3 is re-projected into the right view V_1.

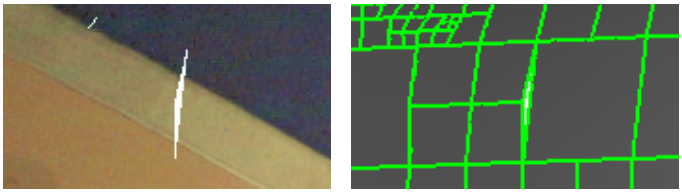


Figure 6.5: The problem of cracks. Cracks appear between quads of different sizes and form a white triangle.

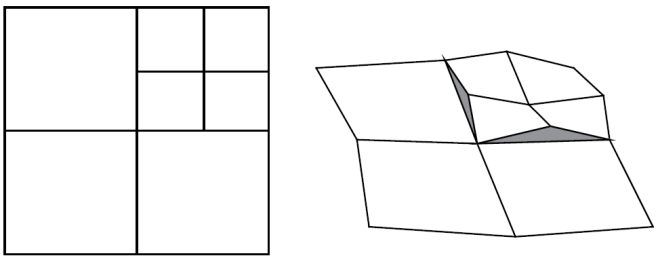


Figure 6.6: Cracks (shaded in grey) appear between quads of different sizes. Figure extracted from [Paj02].

Restriction and triangulation. A method for avoiding these cracks is given in [SS92] and summarized in [Paj02]. It holds in two steps:

1. Restriction of the quadTree subdivision. It consists in subdividing the quadtree such that neighboring quads are within one level of each other in the quadtree hierarchy as shown in figure 6.7.

2. Triangulation. Every quad is triangulated with respect to the size of its neighbor. Due to the constraint of the restricted quadtree hierarchy that the levels of adjacent quads differ at most by one, the quads can be triangulated such that no cracks appear as illustrated in figure 6.8.

Considering the polygon soup representation, an exception of the above method holds when two adjacent quads are separated by a depth discontinuity or when one adjacent quad is discarded from the polygon soup (i.e. after the redundancy reduction). Indeed, in this case the two quads are not connected to each other and no cracks appear but disocclusion areas. Therefore, restriction and triangulation are not applied to such quads.

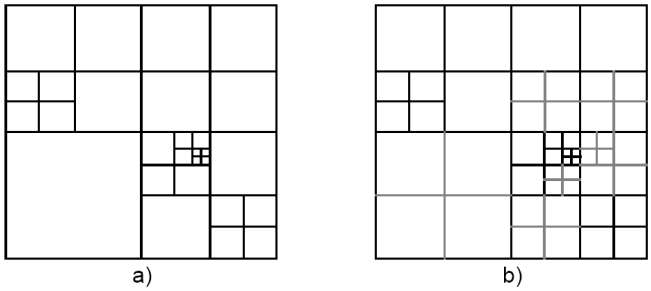


Figure 6.7: Example of an unrestricted quadtree subdivision (a), and restricted subdivision (b). Figure extracted from [Paj02].

Results of cracks elimination. This method of cracks elimination is applied on each quadtree as a pre-process of the view projection step, i.e. at the user side of

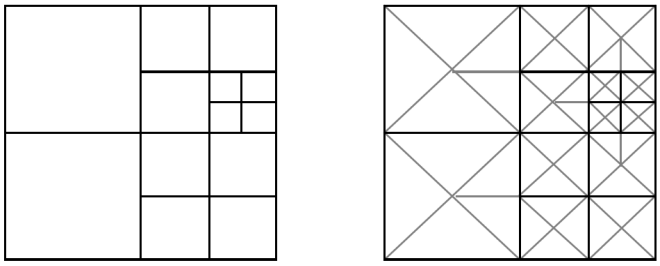


Figure 6.8: Triangulation of a restricted quadtree subdivision. Figure extracted from [Paj02].

the system. Indeed, since this method increases the number of polygonal primitives to be rendered, applying it during the construction of the representation would lower the benefit of the compactness of the representation. Note also that the elimination of cracks is applied only once for each quadtree, on the contrary to the elimination of sampling artifacts that is applied at each view synthesis step when using depth-based view projection.

Figure 6.9 shows the different steps of the crack elimination when applied to a quadtree. In figure 6.9(c), all the cracks have been eliminated.

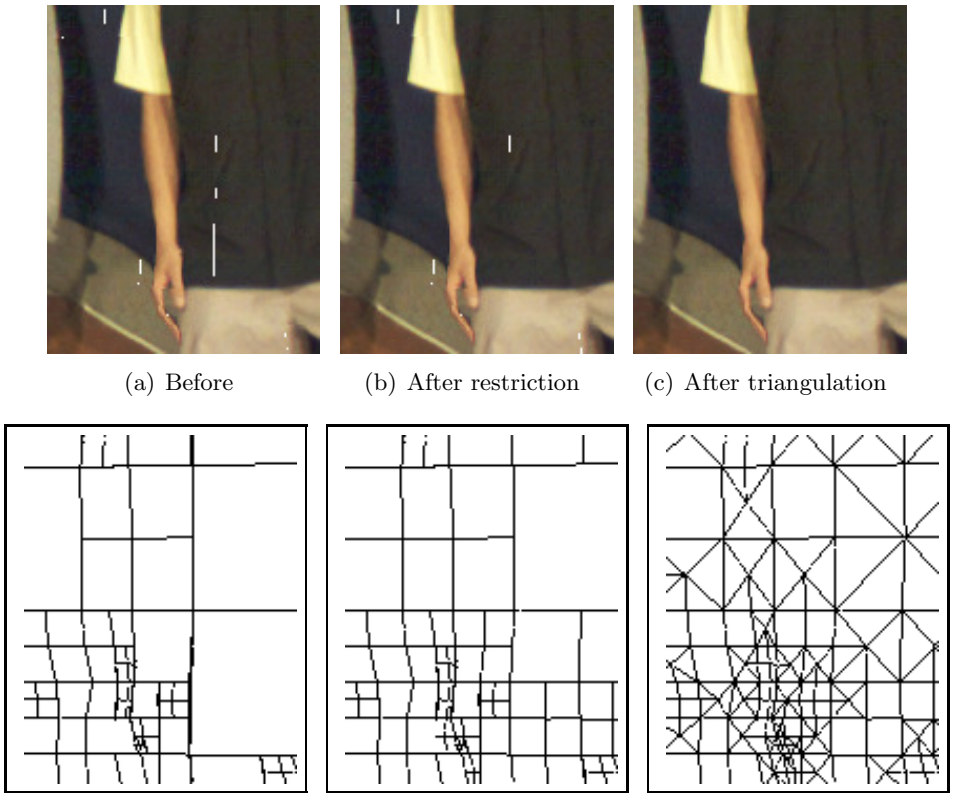
6.2 Multi-view blending

When synthesizing a desired viewpoint, one color image plus geometry information is not sufficient because of occluded areas that become disoccluded in the desired view (as seen in figure 6.4). Instead, using at least two views surrounding the desired viewpoint permits to cover more surface of the scene and greatly reduces disocclusions. Figure 6.10 illustrates this notion. First, both view V_3 and V_1 are projected into the desired view V_2. Large disocclusion areas appear in both images. Then, the two images images are combined into one and disocclusions are greatly reduced.

However, the combination of multiple polygons coming from different views may exhibit some unnatural color changes or small misalignments due to color and geometry inconsistencies across the views. Therefore, a strategy is needed so that textures are correctly combined and views are smoothly interpolated when the viewpoint is changing. The strategy that we present in the following is called adaptive blending 6.11.

6.2.1 Adaptive blending

Color and geometry information may not be consistent across the views. Illumination changes and geometry errors are examples of such inconsistencies. As a result, artifacts may appear in the synthesized view. Figure 6.12 shows a zoomed area of the image 6.10(c) which is a combination of the two projected views V_3 and V_1. In this case, a simple depth test is performed to decide whether the contribution of V_1 or V_3 should be



(a) Before (b) After restriction (c) After triangulation

Figure 6.9: Results of cracks elimination. Cracks are visible before the process (a). The restriction of the subdivision reduces the difference of level between quads to 1 (b). The triangulation eliminates remaining cracks.

(a) P_3^2: Projection of V_3 into V_2

(b) P_1^2: Projection of V_1 into V_2

(c) Combination of P_3^2 and P_1^2

Figure 6.10: Multiview projection: V_3 and V_1 are projected into the desired view V_2 (a) and (b). The combination of this two images greatly reduces the disocclusions.

Figure 6.11: Overview of the multi-view blending stage.

used for each desired pixel. The pixel with smallest depth is the winner. In this figure, unnatural deformations and misalignments are visible around the shoulders and faces of the two characters. This shows that another combination strategy is needed.

Figure 6.12: Artifacts due to inconsistencies between views.

Such a strategy has been first studied in [DTM96] called *View-dependent texture mapping* and in [BBM+01] called *Unstructured lumigraph*. The main idea is to adaptively blend the contributions of different views using criteria such as the camera position, the field of view and the resolution of the cameras. These criteria aim at defining the reliability of a pixel coming from a certain view so that an adapted weight is attributed to this pixel during the blending process. In the following, the blending process is implemented with only the camera position criteria.

Algorithm. The idea is to define a weight to each original viewpoint depending on its distance to the desired virtual view. Using these weights, all the projected views are blended by weighted averaging. Let R_i be the projected view i and D_i its depth map, then the blending process can be formulated as:

$$R(p) = \frac{\sum_i w_i(p) R_i(p)}{\sum_i w_i(p)} \tag{6.1}$$

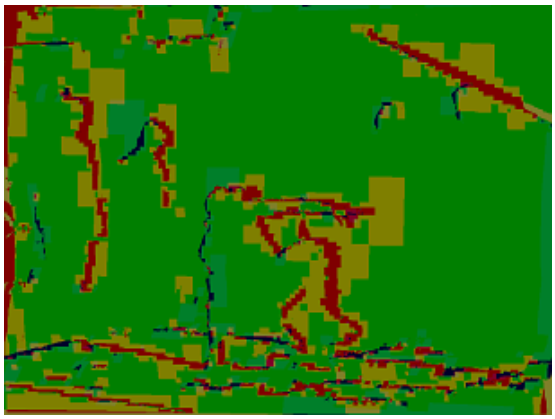
where $w_i(p)$ are weights computed in two steps:

1. for each pixel p, minimum depth values are searched: $D_{min}(p) = \min_i D_i(p)$,
2. $$w_i(p) = \begin{cases} f(dist(curr, i)) & if(D_i(p) < D_{min}(p) + \delta) \\ 0 & otherwise \end{cases} \tag{6.2}$$

The parameter δ is used to robustly reject occluded pixels, and $f(dist(curr, i))$ is a decreasing function of the distance between current viewpoint and viewpoint associated to view i. In our experiment we used $f(dist(curr, i)) = \frac{1}{dist(curr,i)}$.

Results. Figure 6.13 illustrates this process. In sub-figure (d), each view contribution is associated with a color (Red, Green, Blue). These colors are blended when multiple contributions overlap.

(a) V_1 contribution - Red (b) V_3 contribution - Green (c) V_5 contribution - Blue



(d) Weighting factors in virtual V_2

Figure 6.13: Synthesis of virtual view V_2 using adaptive blending. (a),(b) and (c): contributions of other views. (d): relative weighting used for each view (Red component is the weight associated with V_1, Green component is associated with V_3 and Blue component is associated with V_5).

Finally, figure 6.14 compares the combination of two images with the depth test strategy (a) and the blending strategy (b). The blending strategy offers a smoother transition between the quads and reduces the artifacts.

6.2.2 Ghosting artifacts

Ghosting artifacts, or corona artifacts, are strong inconsistencies situated near depth discontinuities. They are either due to geometry errors or color mix between background and foreground. For example, if a background area near depth discontinuity is mixed with the color of the foreground, then the outline of this foreground object will appear in the background during the view synthesis, creating strong artifacts (figure 6.15(a)).

Existing methods for avoiding this kind of artifact consist in extracting discontinuity boundaries, and applying separate process to this area [ZKU+04, SMD+08]. It is performed at the user side and thus increases the view synthesis complexity.

(a) Depth test combination

(b) Blending combination

Figure 6.14: Comparison between the depth test combination of multiple views and the blending combination.

In the polygon soup, ghosting artifacts are automatically eliminated because of the reduction process (section 5.2) that removes unreliable quads. Therefore, there is no need for additional process at the view synthesis stage for removing ghosting artifacts.

The advantage of the redundancy reduction is shown in Figure 6.15. On the left is full polygon soup, i.e. without removing redundant and unreliable quads. The contour of the head clearly appears in the background, creating a ghosting artifact. Small quads are situated in the area of the ghosting artifact. On the right is the reduced polygon soup. As expected, ghosting artifacts are drastically reduced. Quads have been eliminated and replaced by bigger ones coming from another viewpoint.

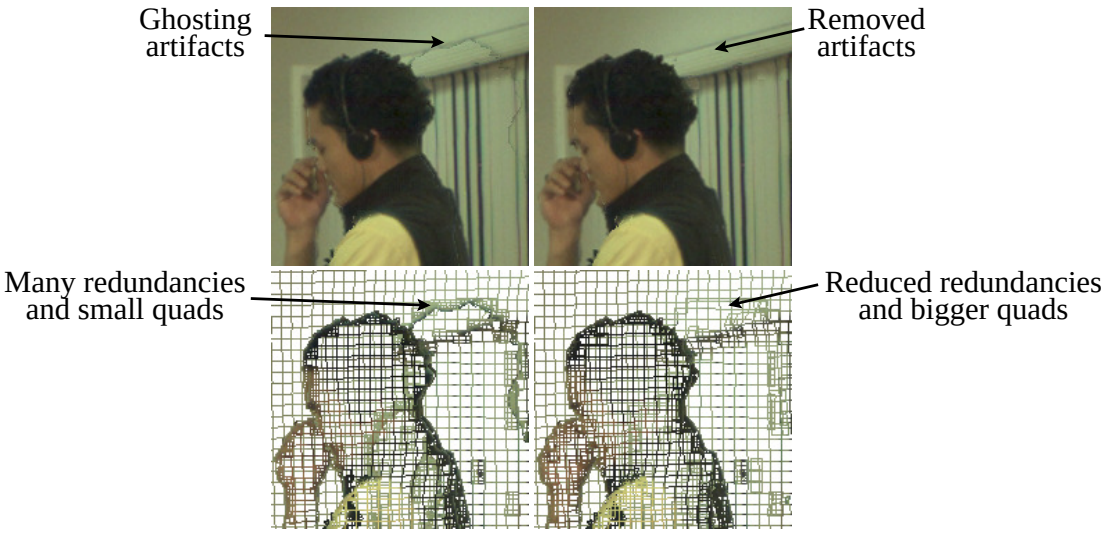


Figure 6.15: Reduction of the ghosting artifact. (left) without redundancy reduction, (right) with redudancy reduction.

6.3 Virtual view enhancement

This section details the enhancement of the virtual views by image processing methods. Synthesized views often contain remaining disoccluded areas (unknown areas) that could not be predicted by any of the original views. A typical hole filling method called 'inpainting' is applied to fill these areas. Finally, an additional filtering process is performed to provide a more natural appearance around object boundaries. These two process are summarized in figure 6.16.

6.3.1 Inpainting

White areas may be observed when synthesizing an intermediate viewpoint. They correspond to non predicted pixels, i.e. pixels that do not appear in any of the available views. These pixels can be easily reconstructed using some inpainting techniques such

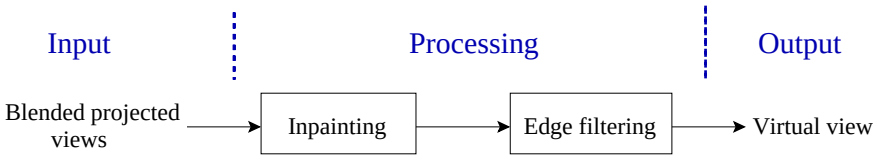


Figure 6.16: Overview of the virtual view enhancement stage.

as [OYH09]. The inpainting method used in this article is the Telea one [Tel04], implemented in openCV: every empty region (i.e. white region with infinite depth) is filled with its surrounding color.

Figure 6.17 shows a detail of the right arm of the breakdancer when virtual view V_2 is synthesized. Sub-figures (a) and (b) illustrate the inpainting process.

6.3.2 Edge filtering

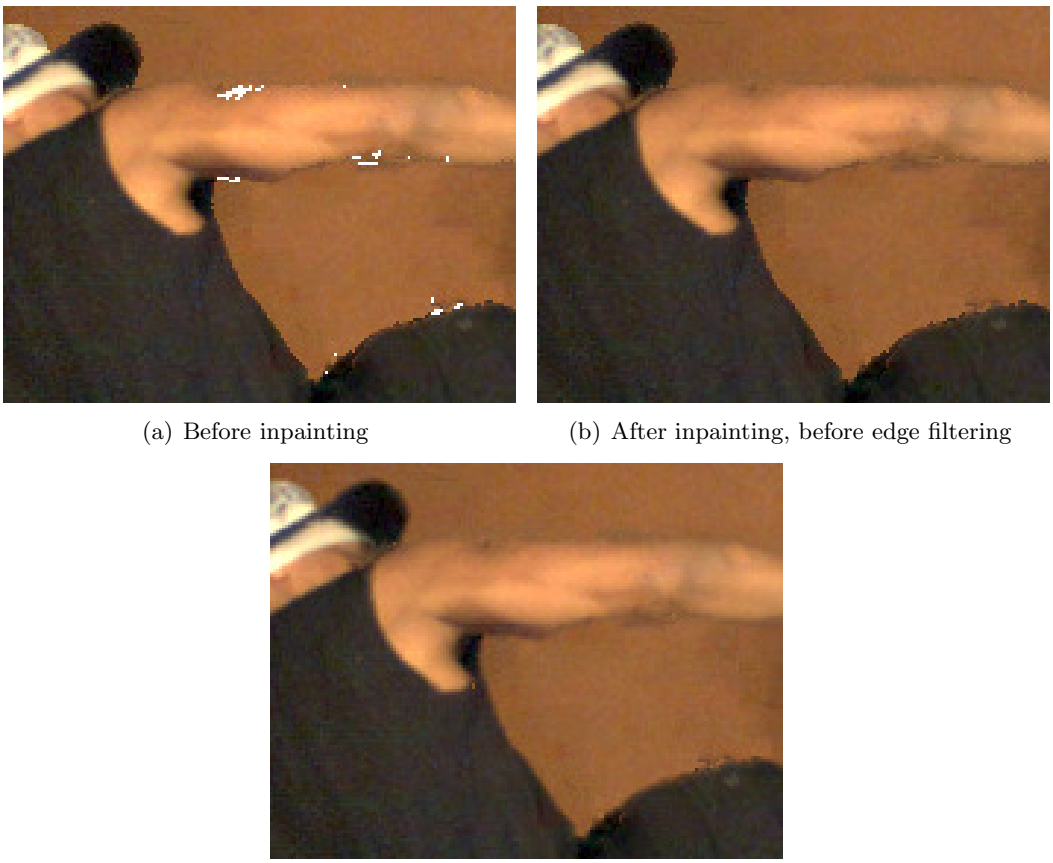
In the original views, object boundary samples are a color mixture of foreground and background objects due to initial sampling and filtering during image capturing. However, when rendering intermediate views, the foreground-background boundaries are changed, resulting in unnaturally sharp edges. Therefore, as in [SMD+08], foreground objects are low-pass filtered along the edges to provide a natural appearance. This filtering also helps to reduce remaining artifacts along depth discontinuities.

Sub-figures 6.17 (b) and (c) show the view synthesis result before and after edge filtering. In (b), the cap and the T-shirt of the dancer look unnaturally sharp in front of the background. After filtering, the appearance is more natural and the irregularities along the leg are less visible.

6.4 Results

In order to evaluate the view synthesis performances, intermediate views were synthesized and compared with the original ones. Figure 6.18 shows the virtual view V_2 for *Breakdancers* and *Ballet* sequences. Figure 6.19 shows the virtual view V_7 for *Book Arrival* sequence.

Subjective quality. These virtual views look realistic and do not exhibit strong visual artifacts. Looking closer, small areas appear distorted or inconsistent as shown in figure 6.20. Figure 6.20 (a) and (b) give a detail of the dancer's face on the right of *Breakdancers*. The right cheek of the dancer is distorted and the bright vertical line in the background is cut instead of being straight. Also the earring of the dancer is reflected in the original view but not in the synthesized view, which shows that specular effects are difficult to reproduce. Figure 6.20 (c) and (d) gives a detail of the background stripes in *Ballet*. Blocky artifacts appear where the stripes are cut instead of being straight. These artifacts appear when the quads originating from different views are not consistent with each other in terms of depth values. This inconsistency



(a) Before inpainting

(b) After inpainting, before edge filtering

(c) After edge filtering

Figure 6.17: Intermediate view: Illustration of the inpainting and edge filtering method used to fill white pixels and to provide a more natural appearance. Contrast has been enhanced for better visualization.

comes from the inaccuracies in the input depth maps and results in projection errors in the virtual views. A solution for reducing these artifacts will be presented in chapter 8.

Objective quality. The objective quality is evaluated on the luminance components on each image with the computation of the PSNR value. Typical values for good quality images vary between 30 to 40 dB or higher.

For comparing the synthesized views with an existing approach based on MVD data, we used the view synthesis software developed by Nagoya University (VSRS 3.0.1) [TFS+08] currently studied in the MPEG-3DV group. The PSNR results computed on 25 frames and 2 virtual views are given in table 6.1.

The average result for *Breakdancers* is 34.5 dB with the quadtree-based approach and 34.2 dB with the MVD approach. For *Ballet*, the PSNR is 34.3 dB with the quadtree-based approach and 35.2 dB with the MVD approach. Finally, for *Book arrival*, the PSNR is 35.9 dB with the quadtree-based approach and 36.9 dB with the MVD approach.

One can notice that the sequence *Book arrival* yields higher PSNR quality than the two others. This can be explained by the fact that the baseline between cameras is shorter and so re-projection errors are smaller. Concerning the polygon soup, a slightly higher PSNR (+0.3 dB) than MVD is obtained for sequence *Breakdancers* and a lower PSNR (-1 dB) is obtained for sequence *Ballet* and *Book Arrival*. It shows that the quadtree decomposition and redundancy reduction steps have slightly decreased the quality of syntheiszed views for sequence *Ballet* and *Book Arrival*.

	Breakdancers	*Ballet*	*Book Arrival*
MVD	34.2 dB	35.2 dB	36.9 dB
Polygon soup	34.5 dB	34.3 dB	35.9 dB

Table 6.1: Comparison of PSNR values computed on 25 frames and 2 virtual views.

6.5 Conclusion

Synthesizing intermediate views raises several issues concerning the geometric primitives, the combination of multiple views, and the processing of remaining artifacts. This chapter has presented the view synthesis stage based on the polygon soup representation. First, the difference of synthesis results between depth-based and polygon-based projections has been highlighted. Depth-based projection creates sampling artifacts that are usually eliminated with a median filtering process. Polygon-based projection exhibit small cracks between quads of different size. An existing method to remove these cracks consists in subdividing the quads before projection. This method has been applied to the polygon soup representation. Second, the combination of overlapping

(a) Virtual V_2 - PSNR = 35.17dB

(b) Virtual V_2 - PSNR = 33.31dB

Figure 6.18: Results of virtual view synthesis.

(a)

Figure 6.19: Virtual view synthesis - View V_7 - PSNR = 36.74dB.

polygons has been detailed. It consists in blending the textures of the overlapping areas once projected into the virtual view. For a view-dependent appearance, the original views are weighted before blending depending on their distance to the desired virtual views. As a result, a smoother transition between the polygons is obtained, and inconsistencies are softened. In addition, results have shown that the adaptive construction of the polygon soup automatically reduces ghosting artifacts. Therefore, there is no need for additional process concerning this kind of artifact. Finally, classical image processing algorithms have been performed to enhance the virtual views. It includes inpainting of unknown areas and edge filtering of object boundaries.

The evaluation of the virtual views has shown that good quality images were obtained, at a PSNR of about 34.5 dB for sequences *Breakdancers* and *Ballet*, and 35.9 dB for *Book Arrival*. However, some artifacts appear: cut of line, blocky artifact, distortions. It is mainly due to geometry inaccuracies or inconsistencies of polygons extracted from different views. A possible solution to reduce these artifacts will be studied in chapter 8.

The complexity of the view synthesis stage is important for real-time visualization and interaction with the users. The different process presented above have not been implemented for real-time performances, therefore no quantitative results are available concerning the complexity. However, a few comments must be made about the complexity. First, one of the constraints that we have defined for the multi-view video system is that it should contain a specialized parallel device such as graphics hardware.

(a) *Breakdancers* - Original View

(b) *Breakdancers* - Synthesized view

(c) *Ballet* - Original View

(d) *Ballet* - Synthesized view

Figure 6.20: Details of virtual view artifacts.

This hardware is optimized for rendering polygons and can perform very efficiently image-array operations providing adapted programming. The proposed view-synthesis method includes quadtree subdivision, projection and texture mapping, blending, inpainting and edge filtering. Compared to view-synthesis methods using depth image-based representation like MVD, a smaller set of polygons replace the points, no median filtering that removes sampling artifacts is needed, neither adaptive processing of edges that removes ghosting artifacts.

Finally, we have seen in figure 6.20 that specular effects like the reflection of the earring are not correctly reproduced for the moment. A possible solution would be to employ a view-dependent texture mapping strategy similarly to Debevec et al. [DTM96] and Buehler et al. [BBM+01]. With these kind of solutions, multiple textures instead of one are used for each polygons, and they are weighted according to the position of the virtual view.

The next chapter is dedicated to the compression of the polygon soup using the quad-tree structure.

Chapter 7

Compression of the polygon soup

Contents

Up to now, the polygon soup is constructed at each time instant with a quadtree decomposition of the input depth maps, and inter-view redundancies are eliminated using an adaptive pruning of the quadtree leaves. Based on this polygon soup and input color images, virtual views can be synthesized within the navigation range with good quality. Transmission over a band limited channel is another issue for multiview video system. In order to evaluate the representation as a candidate for such system, the compression of the representation has to be considered.

This chapter is dedicated to the compression of the polygon soup representation, using the quadtree structure. The compression of the color images used to texture the polygon soup is not considered here. The goal is to reduce the bit rate of the polygon soup using a lossy predictive coding method while avoiding strong artifacts in virtual views. Different parameter values of the polygon soup construction are tested in order to evaluate their influence on the compression performances. Moreover, a comparison with an existing MVD coding scheme is given. This comparison focuses on rate-distortion curves as well as compression artifacts in the synthesized views.

This chapter is organized as follows. First, section 7.1 describes a new quadtree-based compression method adapted to the polygon soup representation. Then section 7.2 tests different settings of the polygon soup in order to evaluate the advantages of the redundancy reduction and the influence of the small quads on the compression performances. Finally, section 7.3 gives the compression performances of the polygon soup compared with an existing MVD scheme.

7.1 Compression method

A quadtree-based compression of the polygon soup is proposed in this section. The input of the algorithm is the polygon soup, i.e. multiple pruned quadtrees for each time instant. The proposed compression method is applied frame by frame such that no temporal prediction is used. Each quadtree corresponding to one view is coded independently.

Quadtree structure. The structure of the quadtree is used to recursively define the position and size of each quad. More precisely, each quadtree is traversed from top to bottom, and for each quad, two flags are used:

- One flag indicates if the quad is activated or if it has been discarded during the redundancy reduction.
- Another flag indicates if the quad is a leaf or if it must be subdivided (node).

Figure 7.1 gives an example of quadtree structure and associated flags. When both flags are positive (activated,leaf) then four depth values are transmitted corresponding to the four corners of a quad.

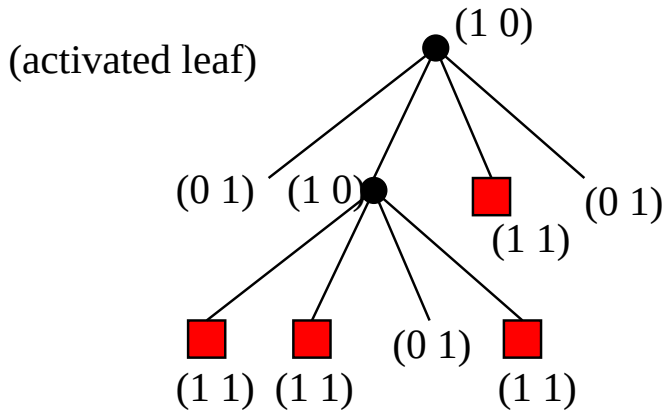


Figure 7.1: Structure of the quadtree and associated flags: (leaf activated)

Depth values. In order to encode the depth values of the selected quads, each corner is predicted using the adjacent quads already coded. Figure 7.2 shows an example of a current quad that can be predicted with already coded quads. The corners of the quads are named with the letters t/b: top/bottom, r/l: right/left.

First, each quad already coded updates a list of predicted depth values for each of its pixels (depth values are interpolated from the quad corners' depth). Since adjacent quads are not necessarily connected to each others, the list can store multiple depth values for the same pixel position. We denote a depth value in the list as $zlist(pixPos, idx)$ where $pixPos$ is the pixel position, and idx is the index of the quad that added the

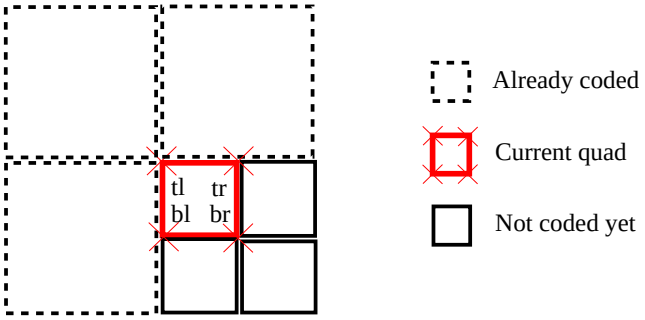


Figure 7.2: Prediction of depth values from already coded quads.

depth value, $idx \in [1, 3]$ i.e. at most three adjacent quads sharing the same corner position can update the list (they are the quads in dotted line in figure 7.2).

Then during the prediction, this list is used to predict the corners of the current quad. More precisely, at a given pixel position, the depth value used to predict the current corner is the one with smallest difference with the current depth:

- $\hat{z}_{tl} = zlist(tl, idx)$ with $idx = \arg\min_i(|z_{tl} - zlist(tl, i)|)$
- $\hat{z}_{bl} = zlist(bl, idx)$ with $idx = \arg\min_i(|z_{bl} - zlist(bl, i)|)$
- $\hat{z}_{tr} = zlist(tl, idx)$ with $idx = \arg\min_i(|z_{tr} - zlist(tl, i)|)$
- $\hat{z}_{br} = \bar{z}_{tr} + \bar{z}_{bl} - \bar{z}_{tl}$. The bottom right corner is predicted using affine prediction from the three other previously coded corners.

where $\hat{z}$ corresponds to a prediction value, $\bar{z}$ corresponds to a decoded depth value (i.e. taking into account codded residue added to prediction). If multiple candidate values are possible for prediction, then the index idx of the chosen value is transmitted with the predicted value hence with an additional coding cost.

This prediction process helps to preserve connections since corners are predicted in priority with the adjacent corner with smallest depth difference.

Because of the deactivation of many quads for redundancy reduction, some quads may not be correctly predicted if neighbors are not activated. In order to still get a valid prediction in this case, when a node is not activated virtual depth values are defined using previously defined predictions. This indeed corresponds to do padding of depth information.

Lossy compression. A prediction residue is then defined to be added to these predictions. Due to high correlation of depth values these residues are often quantized to 0. A flag is then introduced to signal 0 value for these residues. If not zero, then the residue is quantized and ExGolomb code is used to code it. All information is coded using Context Adaptive Binary Arithmetic Coding (as proposed in [MWS03]).

Contexts are established depending on the level of a node in the quadtree to take into account statistical variations among quads size.

7.2 Performance with different settings

For this experiment, the input data and conditions were set as follows:

- The quads of views V_1, V_3, V_5 were compressed, but not the texture images. Therefore, only the bit rate required for the geometry information is studied here.
- Intermediate views V_2 and V_4 were synthesized and compared with the original ones.

Three different settings. We want to evaluate the compression method with different settings of the polygon soup. Three settings were compared. The first one corresponds to the full quadtrees without the redundancy reduction step. Many quads have to be coded with this setting. The second setting corresponds to the quadtrees after the redundancy reduction step. Finally, the third one is also the reduced quadtrees but without any quad of size 1 pixel. Indeed, in the quad representation the number of quads of size 1 pixel is about 30% of the total number of quads whereas the surface covered by these quads in the image is very small. In order to reduce the bitrate the quads of size 1 pixel were not coded at all, so the minimum size of quads is 2 pixel and this setting is refered to as '$QSmin = 2$'.

Results. Figure 7.3 shows the rate-distortion results for the three different settings. These curves are obtained by varying the quantization step of the compression method in order to evaluate different bit rates with different distortions. For the same quantization steps, we can see that the reduced setting reduces the bit rate by at most 0.17 bpp, but the PSNR value is also reduced by 1.5 dB. This shows that the compromise rate vs distortion can be tuned by removing more or less redundancies.

Then, regarding the performance of the 'QSmin=2' setting where quads of size 1 pixel where discarded, we can see that the bit rate is again reduced by 0.04 bpp compared to the reduced setting, while the PSNR quality almost does not decrease. The lack of these small quads are in fact compensated at the view synthesis step thanks to the inpainting and edge filtering process (as explained in section 6).

7.3 Comparative evaluation

Compared approach. The existing approach that is used for comparison is based on the ITU-T H.264 Multi View Coding amendment (MVC version 6.0) [JTC08] for coding the depth maps and the Nagoya university view synthesis software (VSRS 3.0.1) [TFS+08] currently studied in the MPEG-3DV group. In the following, we denote this approach that uses MVD data + MVC compression + VSRS view synthesis as "the

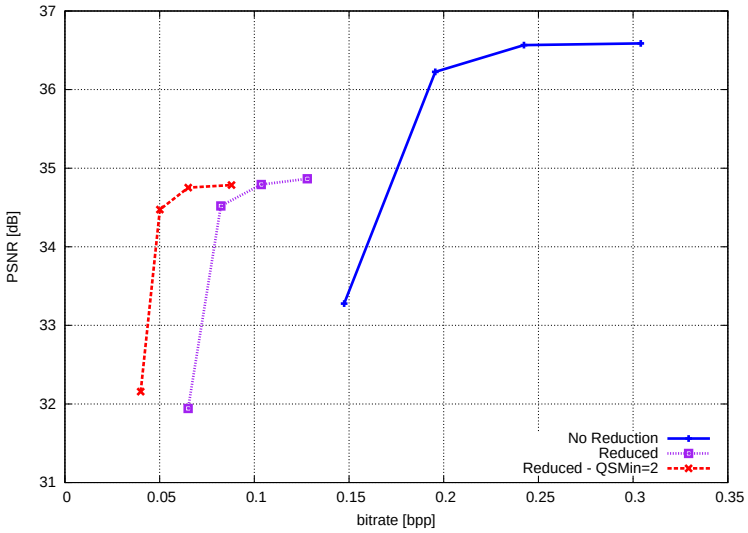


Figure 7.3: Different settings of the polygon soup. Rate distortion performance for view synthesis using polygon soup. Average bitrate VS average PSNR for 2 intermediate views (V_2 and V_4) and 1 frame of *Breakdancers* sequence.

MVD approach". The MVC software is used to compress the depth maps. Inter-view prediction is activated but no temporal prediction is performed because our proposed coding scheme does not take the temporal dimension into consideration at the moment. The central view V3 is coded as I picture, and the two lateral views V1, V5 are coded as P-pictures with central view as prediction (i.e. PIP inter-view coding structure). Four different bitrates and qualities are tested by changing the quantization parameter to typical values: QP = 22, 26, 31, 37. In the VSRS software, the main settings are activated like half-pel precision; bi-linear filtering; boundary noise removal and view blending and inpainting.

Results on geometry. For comparing the geometry of both methods after compression, the compressed quadtrees are converted to depth maps and compared with the compressed depth maps of the MVD approach. Empty areas corresponding to deactivated quads in the quadtrees are not taken into account. Thus, the PSNR is computed between compressed depth maps and original depth maps. Figure 7.4 shows results in terms of rate-distortion performance. The figure shows that the settings ($QSmax = 8$, $QSmin = 2$) outperforms MVC for medium and high bitrates (starting from 0.042 bpp). At most, a gain of 4 dB is obtained at bitrate 0.08 bpp. For low bitrates, MVC outperforms our method. The setting $QSmax = 16$ shows that increasing the maximum size of the quads could provide better results at low bitrates. This indicates that for low bitrates the quadtree structure must be adapted to a coarser approximation.

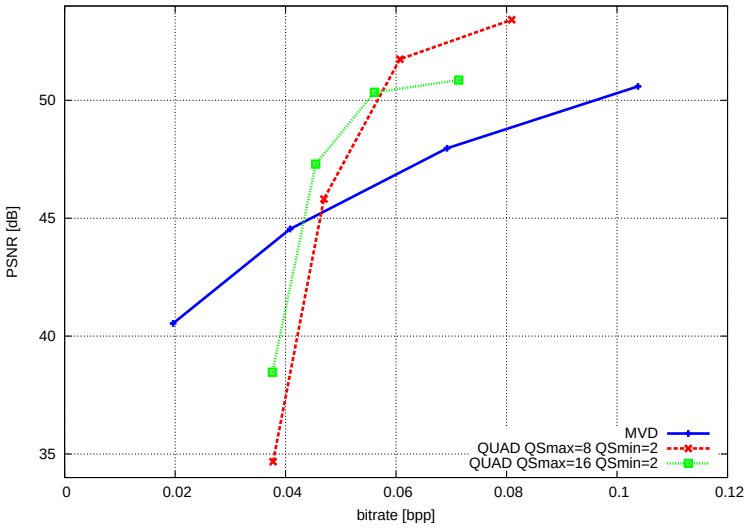


Figure 7.4: Results on geometry. Rate distortion performances for the MVD and quadtree-based methods. Average bitrate VS average PSNR for 3 views and 25 frames of *Breakdancers* sequence.

Results on virtual views. Now the quality of virtual views synthesized with compressed data are compared. The PSNR is computed between the virtual and original views. Figure 7.5 shows the rate-distortions results of both compression methods for sequence *Breakdancers*. The behavior is similar to the previous results on depth. We can see that the quadtree-based approach gives slightly higher PSNR results at medium and high bit rates. At most, a PSNR gain of 0.33dB is obtained at 0.08bpp with the settings ($QSmax = 8$, $QSmin = 2$). Figure 7.6 shows the rate-distortions results of both compression methods for sequence *Book Arrival*. We can see that the quadtree-based approach gives lower PSNR (-1 dB) at medium and high bit rates.

Subjective image quality. We now analyze the subjective quality of the virtual views and the artifacts induced by the two compression methods. Figures 7.7 and 7.8 show a detail example of virtual view V_2 at frame number 13 and 17 respectively. Corresponding videos can be watched at this address [1]. These views were synthesized with MVD approach at bitrate 0.041 bpp and quadtree-based approach at bitrate 0.047 bpp ($QSmax = 8$, $QSmin = 2$). The bit rate difference is small. The original image is also shown in order to compare artifacts induced by the coding and rendering steps.

The MVD approach exhibits white ghosting artifacts around the dancer. On the contrary, the quad based example does not contain such artifacts. In figure 7.8, the red square in the MVD approach highlights a region that has been inpainted during view synthesis using VSRS software. Note that brown color of the background should appear instead of the blurred white color of the dancer's cloth. On the other hand,

[1] http://www.irisa.fr/temics/staff/colleu/

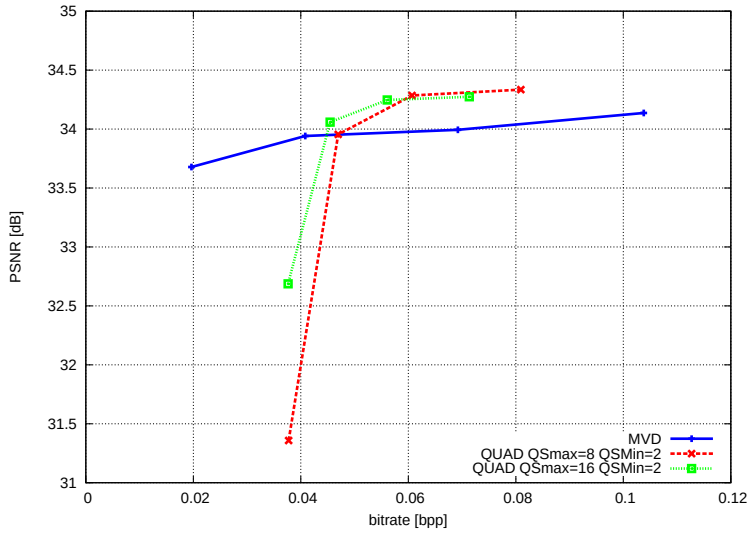


Figure 7.5: Results on virtual views - sequence *Breadancers*. Rate distortion performance for the MVD and quadtree-based methods. Average bitrate VS average PSNR for 2 intermediate views (V_2 and V_4) and 25 frames.

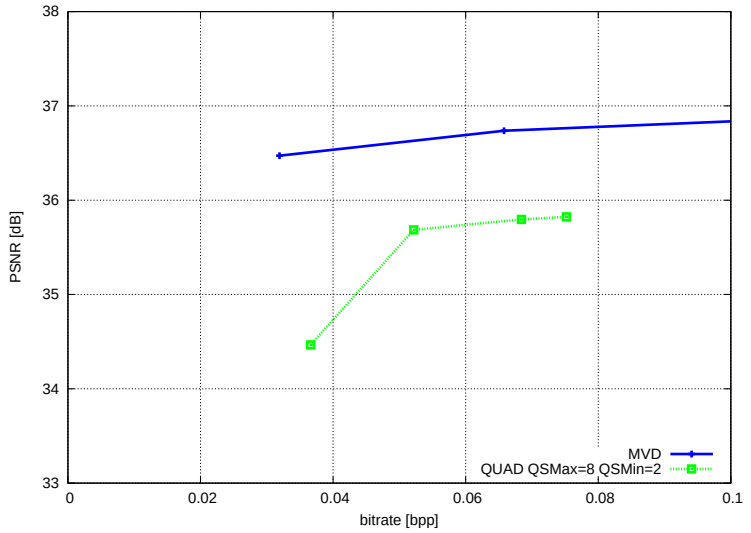


Figure 7.6: Results on virtual views- sequence *Book Arrival*. Rate distortion performance for the MVD and quadtree-based methods. Average bitrate VS average PSNR for 2 intermediate views (V_2 and V_4) and 25 frames.

in the quadtree-based approach, this region does not contain such artifact. In the quadtree-based example of figure 7.8, some distortions appear along the back of the dancer on the contrary to the MVD approach and original image.

We now give the interpretations of the previous observations:

- The ghosting artifacts observed with the MVD approach come from the fact that MVC compression is not designed to preserve depth discontinuities, and therefore typical ringing artifacts appear in the depth maps and finally create ghosting artifacts in the intermediate view. On the contrary, the quadtree-based example does not contain such artifacts because the quadtree structure and compression method avoid ringing artifacts.

- The quadtree-based approach does not contain the inpainting artifact like the one observed with the MVD approach because this region has been filled by the information of view V_5 on the contrary to the MVD approach that used only 2 views for the view synthesis: V_3 and V_1. Using 3 views instead of 2 helps to reduce the size of unknown areas in the intermediate views.

- The dancer's boundaries with the quadtree-based method in figure 7.8(c) are distorted because of the compression step. Although quads are predicted in priority from connected neighboring quads, this is not always ensured since quads are predicted from their top-left neighboring quads which are not always connected. If a quad is predicted with a disconnected corner, then high prediction residue are quantized and create high errors.

These results show that the proposed quad-based representation enables to reconstruct good quality virtual views. It outperforms the MVD approach at medium and high bitrates in terms of objective quality measures. It provides equivalent overall visual quality, with no ghosting artifacts.

7.4 Conclusion

In this chapter, an adapted compression technique of the polygon soup has been presented. The method takes advantage of the quadtree structure that helps to efficiently retrieve the position and size of each quad. The compression method consists in predicting the depth values of the quads' corners using the already coded neighbor quads. Then, prediction residues are quantized and coded with context adaptive binary arithmetic coding (CABAC).

The evaluation of the compression method was split in two. First, different settings of the polygon soup have been compared. It has shown that the redundancy reduction step clearly reduces the bit-rate (about 0.1 to 0.2 bpp reduction) but also reduces the PSNR (about 1.5 dB reduction) in the synthesized images. Therefore, the compromise between rate and distortion can be tuned by removing more or less redundancies. Moreover, the configuration without the quads of size one pixel results in about 0.04

(a) Original

(b) MVD 33.5 dB

(c) Quad-based 34 dB

Figure 7.7: Subjective image quality on frame 13. Zoom on virtual view obtained with MVD and quadtree-based approach at equivalent bit rates.

(a) Original

(b) MVD 33.1 dB

(c) Quad-based 33.6 dB

Figure 7.8: Subjective image quality on frame 17. Zoom on virtual view obtained with MVD and quadtree-based approach at equivalent bit rates.

bpp reduction without decreasing the PSNR. This shows that these small quads have a high cost in rate compared to their influence on the image quality.

In the second experiment, the performances of the compression method have been compared with an existing approach that uses multi-view plus depth representation (MVD) compressed with multi-view coding technique (MVC). Results have shown that the proposed method gives good performances at medium and high bit rates. Moreover, no ghosting artifacts were observed with the quadtree-based method, whereas these artifacts appear when using the compressed MVD representation.

The compression method studied in this chapter has shown that it is possible to efficiently compress the polygon soup representation proposed in this thesis. The MPEG's 3DV group is currently studying how to represent and compress the depth information of MVD data. Most of the envisioned solutions use an image-based compression method adapted to preserve the depth discontinuities. The proposed polygon soup and associated compression technique could be considered as an alternative to these methods. However, the compression method is not yet mature, and more improvements are necessary to reduce distortions: first, the method is not efficient at low bit rates. An optimization of the parameters of the quadtree during the quadtree decomposition could help improving the performances at such bit rate. The thresholds that control depth discontinuities, geometry accuracy and maximum quads' size can be tuned for this purpose. Second, the object boundaries exhibit small distortions due to the compression step. An adaptation of the compression method is necessary. A possible improvement could be to modify the parsing of the quads during the prediction step. Indeed parsing the quads from the biggest size to the smallest size could help predicting smallest quads around discontinuities at last, thus prediction from connected neighbors quads and not only from top left neighbors quads would be possible.

The consideration of the temporal dimension is still to be investigated for our compression method. Similarly to the video compression approach in Urvoy's work [UCP+09], the representation is defined as a set of polygons. The extension to the temporal dimension could consider the temporal evolution of a polygon. A frame would then be reconstructed by sets of polygons coming from different viewpoints (as already seen) but also from different temporal time instant. This notion of evolution of the polygons leads to a dynamic version of the polygon soup. In the next chapter, the evolution of the polygons through space is presented. It aims at reducing texture misalignments and distortions in the synthesized views.

Chapter 8

Floating geometry

Contents

In chapter 6, we have shown that good quality virtual views can be synthesized using a polygon soup representation. However, some misalignment artifacts appear in synthesized views. In this chapter we will introduce a novel method called *floating geometry* that aims at reducing these artifacts.

First the problem will be explained in more details in section 8.1, then existing solutions will be given in section 8.2. Section 8.3 is dedicated to the principle of floating geometry and finally section 8.4 will give the results when floating geometry is applied on polygon soup.

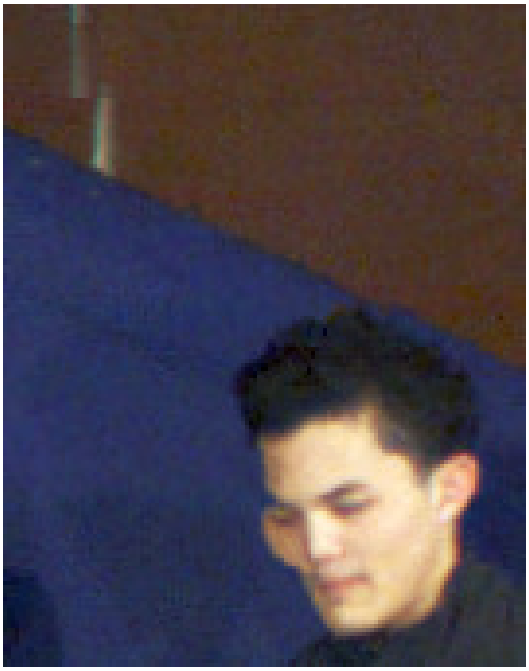
8.1 Texture misalignments

This section describes the texture misalignments issue. First, texture misalignments and their resulting artifacts in synthesized views are introduced. Then the causes for these misalignments are explained in order to better understand how they can be corrected.

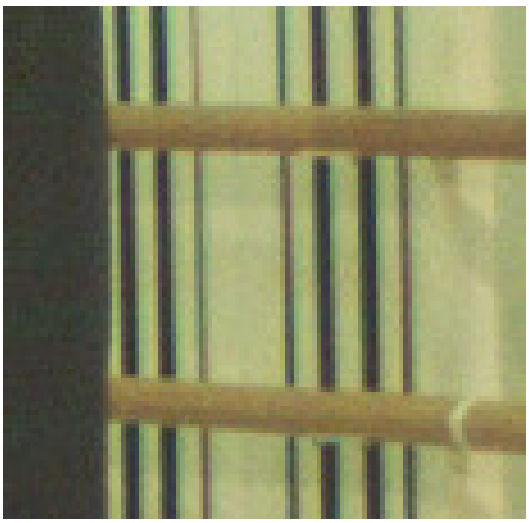
Generally speaking, texture misalignments correspond to position errors of a texture after the view synthesis process. It usually results in visible artifacts in the synthesized view. Figure 8.1 shows synthesized views obtained with a polygon soup representation. Misalignments appear in the form of deformation of important details (e.g. face); cut

or misalignments of structured textures (e.g. straight lines); or blur and ghosting when multiple misaligned textures are blended together.

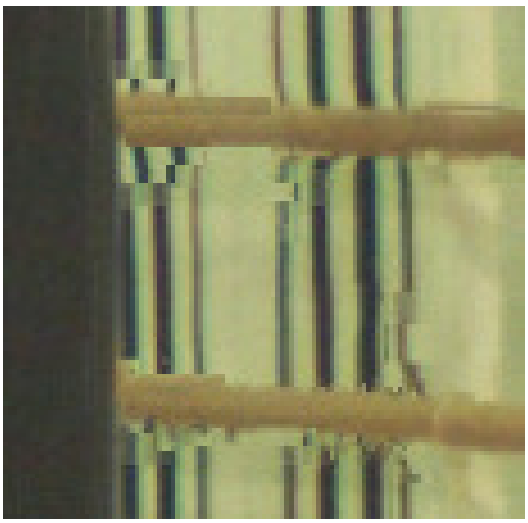
(a) *Breakdancers* - Original View



(b) *Breakdancers* - Synthesized view



(c) *Ballet* - Original View



(d) *Ballet* - Synthesized view

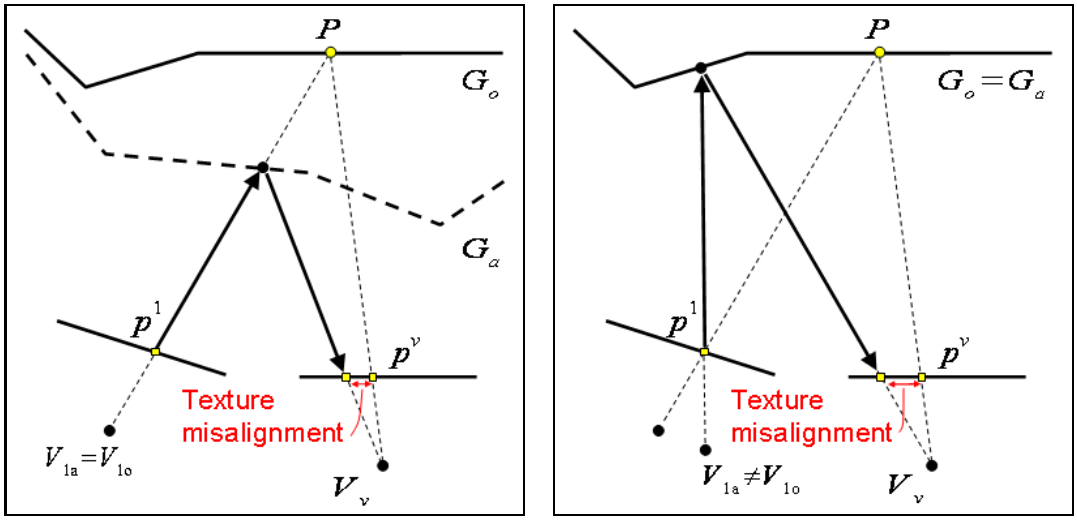
Figure 8.1: Details of artifacts due to texture misalignments: deformation of the face (b), blur where multiple textures are blended (d), and cut of straight lines (b and d).

Modeling errors. During the view synthesis process, models of geometry and camera are used. They are both estimated from the multiple input views and may contain inaccuracies. Indeed, although huge progress have been made in this field, real world scenes are still difficult to model and some typical situations still lead to reconstruction errors and camera calibration errors. The estimation method usually relies on matching features or regions across multiple views, according to a certain similarity measure. False match can occur for example in areas with low or repeated textures, or in reflecting areas. Moreover, both camera and geometry are defined by parametric models and the parameters are estimated by minimizing an error over a high number of data, thus errors are usually non zero. All these uncertainties result in geometry and camera parameters that approximate the observed data, but contain residual errors.

Texture misalignments. The residual errors contained in the camera parameters or geometry result in texture misalignments after the view synthesis process. This is now explained in more details. Figure 8.2 shows the synthesis of a virtual view V_v using input view with approximate camera parameters V_{1a} and approximate geometry G_a. The view with original camera parameters is V_{1o} and the real-world geometry is G_o. Sub-figure (a) illustrates the texture misalignments due to geometry errors only ($G_a \neq G_o$; $V_{1a} = V_{1o}$). The 3D point P is captured in V_{1o} at pixel position p^1. Then during the view synthesis stage (illustrated with arrows), p^1 is back-projected on G_a and re-projected in V_v. The resulting pixel is not aligned with p^v which is the direct projection of P into V_v. This shows that the geometry errors in G_a cause texture misalignments. Sub-figure (b) illustrates the texture misalignments due to camera errors only ($G_a = G_o$; $V_{1a} \neq V_{1o}$). During the view synthesis stage (illustrated with arrows), p^1 is back-projected on G_a using the approximate camera parameters V_{1a} and re-projected in V_v. As previously, the resulting pixel is not aligned with p^v which shows that camera errors in V_{1a} cause texture misalignments.

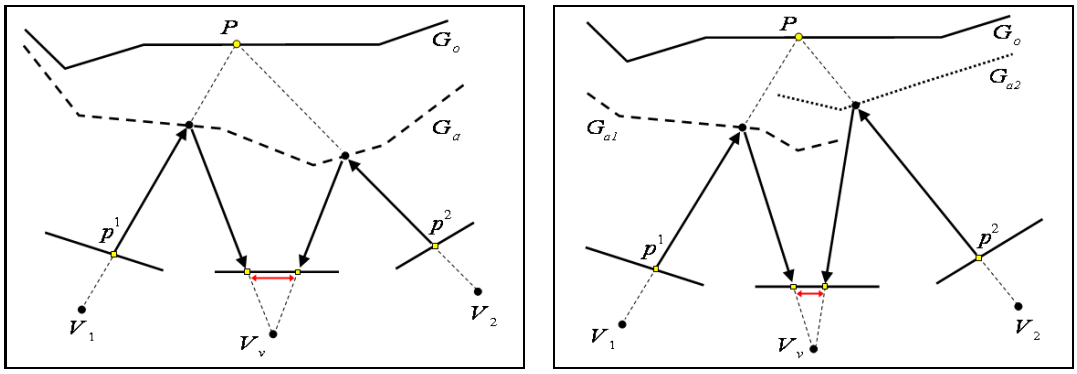
In order to simplify the upcoming explanations and illustrations, we will consider in the following only the texture misalignments due to geometry errors. The same algorithms can then be similarly applied to texture misalignments due to camera errors.

Single and multiple geometries. Existing geometry representations used for 3D video may consist in single or multiple geometries. In the single one, a unique geometry is reconstructed using all the views, resulting in a global error. Figure 8.3(a) illustrates the view synthesis stage using a single geometry G_a and two views V_1 and V_2. Texture misalignments appear due to the global geometry error. On the other side, multiple geometries are reconstructed from a subset of views so that each view is associated with a different geometry, like in the multi-view plus depth (MVD) representation. As a result, the geometries may not be consistent with each other and the superposition of such geometry results in texture misalignments. Figure 8.3(b) shows the view synthesis with two multiple geometries G_{a1} and G_{a2}. Texture misalignments appear due to local errors and geometry inconsistencies. Therefore, both single and multiple representations suffer from texture misalignments.



(a) Geometry errors (b) Camera parameters errors

Figure 8.2: Texture misalignments due to geometry errors (a) and camera parameters errors (b). The approximate geometry is G_a and approximate camera is V_{1a}. The 3D point P is captured by the cameras at pixel positions p^1 and p^v. The arrows illustrate the view synthesis process. The re-projection of p^1 into V_v is not aligned with p^v which shows the texture misalignment.



(a) Single geometry (b) Multiple geometries

Figure 8.3: Texture misalignments with single geometry (a) and multiple geometries (b). The arrows illustrate the view synthesis process. In (b), the approximate geometry G_{a1} is associated with view V_1 and G_{a2} is associated with V_2. In both case, geometry errors cause texture misalignments.

View-dependent texture misalignments. The extent of the misalignment depends not only on the amount of error in G_a but also on the position and orientation of the virtual view V_v. For example, the closer V_v is from V_1, the shorter the misalignment, until the misalignment disappears when V_v is equal to V_1. This property will be exploited in the following such that closer views are more important than further ones: views will be weighted according to their distance to the synthesized view.

8.2 Existing solutions

This section presents recent contributions that deal with the problem of texture misalignment in virtual views in the context of multi-view video. The first one was introduced by Eisemann et al. [EDS+08] and is called *floating textures*. The second one is a modification of the floating textures presented in the same paper. We call it *warped texture coordinates*. These two solutions can be thought as texture registration methods where the first one is performed during the view synthesis whereas the second one is performed as a pre-process of the view synthesis. A different method was proposed by Furihata et al. [FYT+10] where the texture misalignment is not treated as a position error but as an intensity error. We call it *residual error feedback*. Finally, the Ph.D. thesis [Bal05] realized by Balter deals with geometry deformation for reducing texture misalignments.

Floating textures [EDS+08]. The floating textures method has been proposed by Eisemann et al. for correcting the misalignment of the textures during the view synthesis process. This method is explained using a single geometry representation, but can also be applied when using multiple geometries. The idea is to compute the optical flow between synthesized textures coming from multiple views and then use the estimated motion flow to warp them onto each other into an intermediate position so that to reduce texture misalignments. Thus it can be thought as a texture registration method. Since the desired virtual view may be closer to certain views, an angular weighting scheme is used to take into account the position and orientation of the virtual view with respect to the original views during the correction of the textures.

Following the notations of Eisemann et al., each view V_i is projected into the desired viewpoint V_v resulting in the image I_i^v. The computation of the optical flow between two synthesized images gives the flow field $W_{I_i^v \to I_j^v}$ from image I_i^v onto image I_j^v. Therefore, the combined flow field for an image I_i^v is given by:

$$W_{I_i^v} = \sum_{j=0}^{n} \omega_j W_{I_i^v \to I_j^v}$$

where ω_j is the weight attributed to I_j^v, and n is the number of views used for the synthesis. The weight ω_j for view V_j is inversely proportional to the angle between the viewing directions of V_j and V_v, as proposed in [BBM+01, DTM96]. The final image is obtained by warping each image with its combined flow field and summing the resulting images:

$$I^v_{Float} = \sum_{i=0}^{n} (W_{I^v_i} \circ I^v_i)\omega_i$$

where $W_{I^v_i} \circ I^v_i$ warps image I^v_i. Figure 8.4 shows the principle of the floating textures on the same example as previously using a single geometry for all the views. Thanks to the computation of the optical flow, the re-projected pixels p^v_1 and p^v_2 are matched together and then warped one onto each other into an intermediate position using the combined flow fields $W_{I^v_1}$ and $W_{I^v_2}$.

This algorithm ensures a perfect alignment of textures if the computed flowfields are invertible, i.e. $W_{I^v_i \to I^v_j} \circ W_{I^v_j \to I^v_i} = Id$. Indeed, in this case it ensures a perfect warping: $W_{I^v_i \to I^v_j} \circ I^v_i = I^v_j$. However, in practice, this condition is not exactly realized and therefore some misalignments may remain. Concerning the complexity, for n views, it requires computing $(n-1)n$ flow fields for any desired virtual view which is computationally expensive. In practice, it often suffices to consider only the 3 closest views to the desired virtual view.

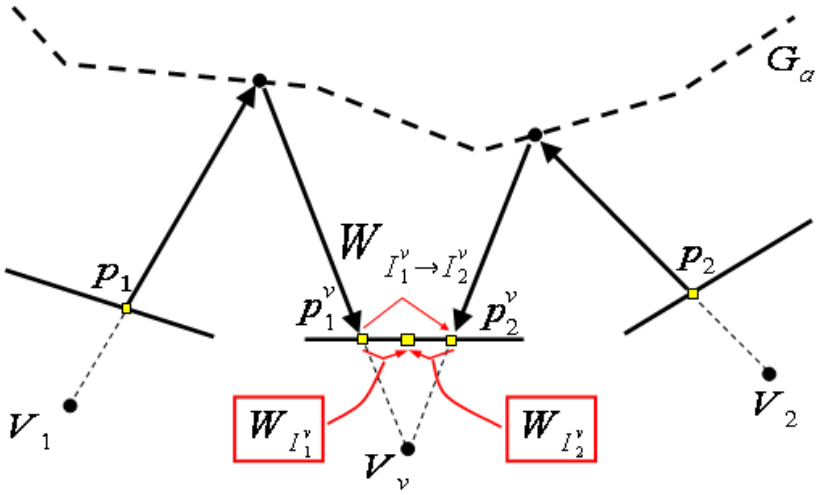


Figure 8.4: Floating textures: p^v_1 and p^v_2 are matched together and then warped one onto each other into an intermediate position using the combined flow fields $W_{I^v_1}$ and $W_{I^v_2}$.

Warped texture coordinates [EDS+08]. In order to reduce the complexity, a variation of the floating textures was proposed in the same paper. The idea is to compute the texture correction as a pre-processing step directly in the original views.

For a given view V_i, all other views are projected into it, resulting in images I^i_j, and the flow fields $W_{I_i \to I^i_j}$ are established. Then during virtual view synthesis, the flow fields of a view V_i are combined using the same angular weighting scheme as in classical floating textures:

$$W_{I_i} = \sum_{j=0}^{n} \omega_j W_{I_i \to I_j^i}$$

This combined flow field W_{I_i}, computed with respect to the virtual view V_v, is applied to the texture coordinates in image I_i of each 3D vertex. Therefore this method is called warped texture coordinates. Note that the texture itself is not warped but only the texture coordinates of each vertex. The final virtual image I_{out}^v is obtained by combining all the images synthesized using the warped texture coordinates:

$$I_{out}^v = \sum_{i=0}^{n} (W_{I_i} \circ I_i)^v \omega_i$$

where $(W_{I_i} \circ I_i)^v$ is the projection of I_i into V_v using warped texture coordinates.

Figure 8.5 shows the texture coordinates of 3D vertex P_a in views V_1 and V_2. Before computing the warped texture coordinates, P_a is associated with two texture coordinates t_1^1, t_2^2, i.e. one in each view. However, because of geometry errors, it is clear that these two texture coordinates do not correspond to the same color. Therefore texture coordinates are warped depending on the desired viewpoint. After computing the warped texture coordinates, P_a is then associated to four texture coordinates: (t_1^1, t_1^2) for synthesizing view V_1, and (t_2^2, t_2^1) for synthesizing view V_2.

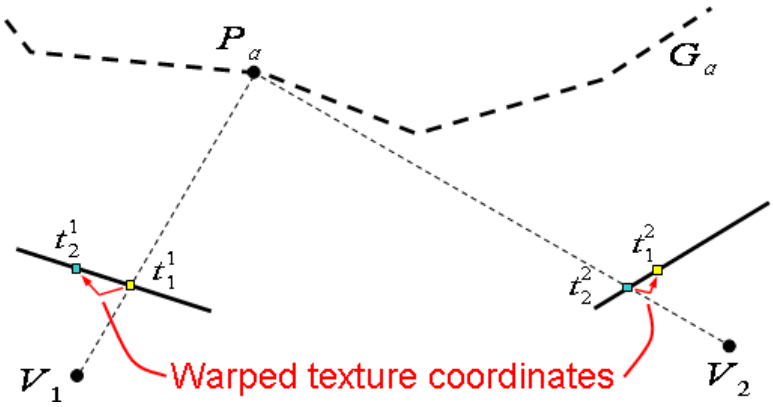


Figure 8.5: Warped texture coordinates: before warped texture coordinates, the 3D point P_a is associated with two texture coordinates (t_1^1, t_2^2). After, P_a is associated with four texture coordinates: (t_1^1, t_1^2) for projection in V_1, and (t_2^2, t_2^1) for projection in V_2.

Figure 8.6 shows the view synthesis step of warped texture coordinates. The two views V_1 and V_2 are used to synthesize V_v. The vertex P_a is projected in V_v. Warped texture coordinates are used for P_a instead of using original texture coordinates t_1^1 and t_2^2. Since V_v is situated in between the two views, then the warp is weighted according to this position and an intermediate texture coordinate between t_1^1, t_2^1 for V_1 and t_1^2, t_2^2 for V_2 is obtained.

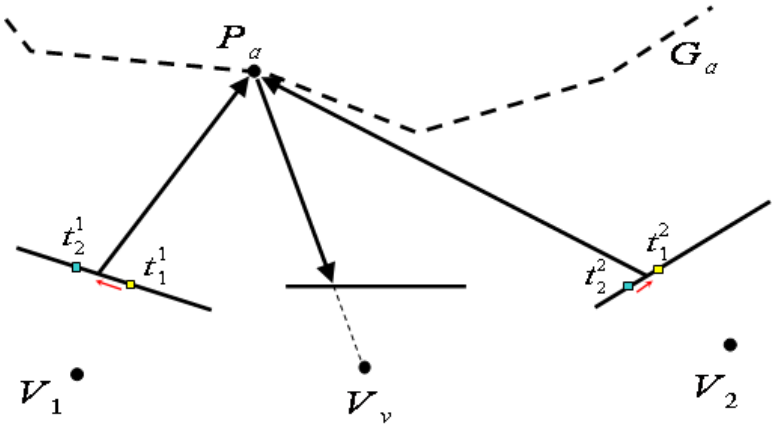


Figure 8.6: Virtual view synthesis with warped texture coordinates. The texture coordinates are warped according to the position of the virtual view. The warping is shown by the arrows newt to the image planes.

The warped texture coordinates require the computation of $(n-1)n$ flow fields at the pre-processing step. During the view-synthesis, only simple warping of the texture coordinates is needed which is computationally efficient. Although this variation of floating textures helps reducing the computational complexity, the quality of synthesized view may be reduced compared with the classical floating textures since the texture correction is not computed in the desired view but indirectly in the original views.

A similar method called *harmonised texture mapping* was proposed by Takai et al. [THM10]. In this method, the computation of warped texture coordinates is reduced to only the vertex positions instead of computed flow fields over the wholes images. The warped texture coordinates are computed as a pre-processing step together with a mesh optimization process.

Residual error feedback [FYT$^+$10]. Furihata et al. proposed another solution to compute the texture correction in a pre-processing step. In this paper, the cameras are rectified on a horizontal axis and a virtual view is synthesized using the two closest lateral views. Thus, when synthesizing the virtual view V_v, then the left view V_l and right view V_r are used.

The pre-processing step of the method consists in computing the intensity residual error of each view projected into the other one. During view synthesis, the residual error is fed back into the virtual view and processed according to the position of this virtual view such that the error increases in proportion to the camera distance. Taking the example of the left view, the synthesized image compensated by the residual error feedback can be formulated as:

$$\bar{I}_l^v = I_l^v + \phi^v(\epsilon_l^r)$$

where ϵ_l^r is the residual error of the projection of V_l into V_r and ϕ^v is a function that processes the error depending on the position of V_v.

The main difference with the previous methods is that the texture misalignment is computed in terms of intensity error instead of flow field which is computationally less expensive. Another difference is that the projections are reversed compared with the previous method: the correction of V_l is computed by comparing I_l^r and I_r (outgoing projection), whereas in the warped texture coordinates, the correction of V_l is computed by comparing I_l and I_r^l (ingoing projection).

Geometry deformation [GBMD04] and 3D scene flow [VBK05]. Galpin and Balter [GBMD04, GBMP04, Bal05] proposed a deformation of the geometry in order to reduce texture misalignments. Multiple geometrical models are used. In order to reduce texture misalignments, the vertices of each model are matched and morphed onto each other at intermediate positions, during virtual view synthesis. This geometry morphing is done across time and can be generalized across multiple views in our context. On the contrary to the above contributions, the match between vertices is not guided by the texture but by the geometry. The problem of matching geometry with texture consistency has been studied for example in [ZBVH09]. In this work, a 3D feature detector and 3D feature descriptor are introduced for matching and tracking surfaces over temporal sequences.

Concerning geometry deformation across time, Vedula et al. [VBK05] introduced the *3D scene flow* with a volumetric geometry representation. At each time instant, input views and one single volumetric geometry are available. The 3D scene flow gives the motion of each voxel of the representation between two frames of the video. It is computed using 2D optical flow and 3D data.

8.3 Principle of floating geometry

This section presents our proposed method for reducing texture misalignments in synthesized views called *floating geometry*.

Overview. The idea of the floating geometry is to deform the geometry in order to keep a consistent geometry over space and time and to reduce texture misalignments. Indeed, since neither a single geometry nor multiple geometries are free of texture misalignments in virtual views, we propose to deform the geometry for each virtual view. Therefore, the goal is not to estimate the real geometry of the scene, but rather to adapt the geometry for each desired view.

Floating geometry is first computed between every original views, as a pre-process before the view synthesis step. This computation uses optical flow estimation and back-projection. Then, for synthesizing a virtual view, the geometry is floated to intermediate positions using the values pre-computed in order to adapt to the virtual view. Concerning the texture, all views projected into the virtual view are blended to allow for smooth combination, as described in section 6.2.1.

This method is quite close to the warped texture coordinates method: texture misalignment is computed using optical flow estimation and this is done between original views as a pre-process. However, the geometry is deformed, instead of texture coordinates. Floating geometry is also close to the 3D scene flow method. In both methods optical flow is used to guide a 3D deformation from one geometry onto another one. However, floating geometry is introduced for correcting texture misalignments whereas 3D scene flow focused on time interpolation.

Principle. Starting from n original views, each view V_i is associated with a geometry G_i and color image I_i (if a single geometry G is used, it is duplicated $n-1$ times such that each view is associated to one geometry G_i).

Floating geometry is first computed as a pre-process in original views: for each view V_i and for all other views V_j, floating geometry G_i^j is computed such that it is consistent with G_j and such that the projection I_i^j of V_i in V_j is aligned with I_j (i.e. alignment of textures). This can be formulated as follows:

$$G_i^j = \pi_{G_j}^{-1}(W_{I_i^j \to I_j} \circ I_i^j)$$

where $W_{I_i^j \to I_j}$ is the flow field between I_i^j and I_j and $\pi_{G_j}^{-1}$ is the back-projection operator using geometry G_j. This flow field is used to warp I_i^j, and the resulting image is back-projected in 3D using the geometry G_j in order to obtain the floated geometry G_i^j.

To summarize, the aim of this process is to obtain a floating geometry G_i^j that is consistent in 3D with G_j while reducing texture misalignments between I_i^j and I_j. In practice, not all the pixels are back-projected but only those corresponding to the vertices of the geometry as illustrated in figure 8.7. In this example, two geometries G_1 and G_2 are associated with original views V_1 and V_2 respectively. It illustrates the computation of the floating geometry G_1^2 that aligns I_1^2 with I_2: the vertex P_1 with texture p_1 is projected in V_2 giving p_1^2. This pixel is matched and warped onto the pixel p^2, and finally this pixel is back-projected onto the geometry G_2 giving the floating vertex P_1^2. Thus when synthesizing view V_2, the vertex P_1 with texture p_1 is floated to vertex P_1^2 which is therefore consistent with G_2 and then projected into V_2. This shows that texture misalignments in original views can be reduced and that multiple geometries can be floated onto each others using floating geometry.

Once floating geometry G_i^j has been computed for all views in a pre-processing step, then it has to be defined for any virtual view during the view synthesis stage. The key point here is to ensure texture alignment by keeping the geometries consistent with each others, thus we want to compute an intermediate geometry G_i^v that depends on the position of the virtual view. Let V_i be the current view to be projected into V_v. The floating geometry for each vertex has already been computed for the original left view V_l and right view V_r around V_v, namely G_i^l and G_i^r. Then the computation of G_i^v consists in interpolating the geometry between G_i^l and G_i^r, vertex by vertex:

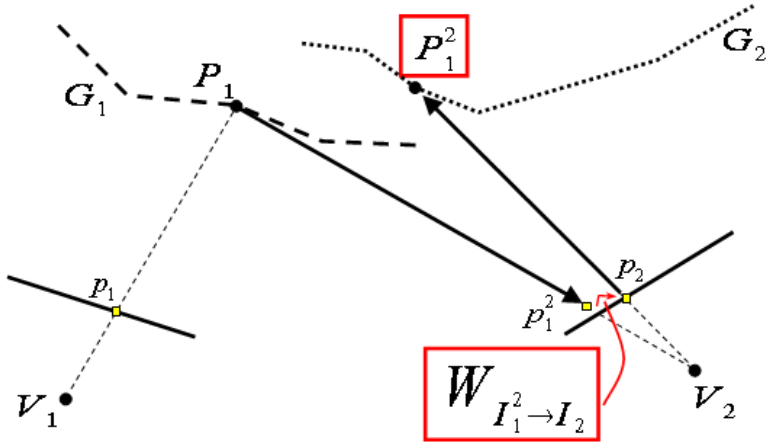


Figure 8.7: Computation of a floating vertex. P_1 is projected in V_2 giving p_1^2 which is matched with corresponding pixel p_2. Finally, p_2 is back-projected to the approximate geometry resulting in the 3D point P_1^2.

$$G_i^v = \omega_l G_i^l + \omega_r G_i^r$$

where ω is a weight defined by similar weighting scheme as previous methods: ω for a view V is inversely proportional to the angle between the viewing direction of V and V_v. Because of this interpolation, a smooth deformation of the geometry is obtained as the virtual view moves from one original view to another. Figure 8.8 illustrates this view synthesis step using the same example as in the precedent figure. The view being projected is $V_i = V_1$ and the left and right views around the virtual view are $V_l = V_1$ and $V_r = V_2$. The 3D vertex P_1 is floated onto an intermediate position $P_1^v = \omega_1 P_1 + \omega_2 P_1^2$. Finally, P_1^v is projected into the virtual view, giving the pixel p_1^v.

This process is repeated for each view and associated geometry as illustrated in figure 8.9 where G_1 and G_2 are floated to G_1^v and G_2^v respectively. As a result, the geometries are consistent with each other, and texture misalignments are reduced both when synthesizing original views and virtual views.

Pros and cons. We have seen that floating geometry aims at reducing texture misalignments. To do so, the geometry is deformed onto a position guided by motion estimation between synthesized and original images. For each view, one geometry is used, and all the geometries are deformed to be consistent with each other for any virtual view. This method can be used for both single or multiple geometries representations.

One advantage of floating geometry, as described above, is that it can be extended to the temporal dimension so that a consistent geometry can be modeled across time and potentially an efficient temporal coding scheme can be designed.

However, similarly to the floating textures and warped texture coordinates presented previously, a perfect texture alignment is not guaranteed. Indeed the computed flow

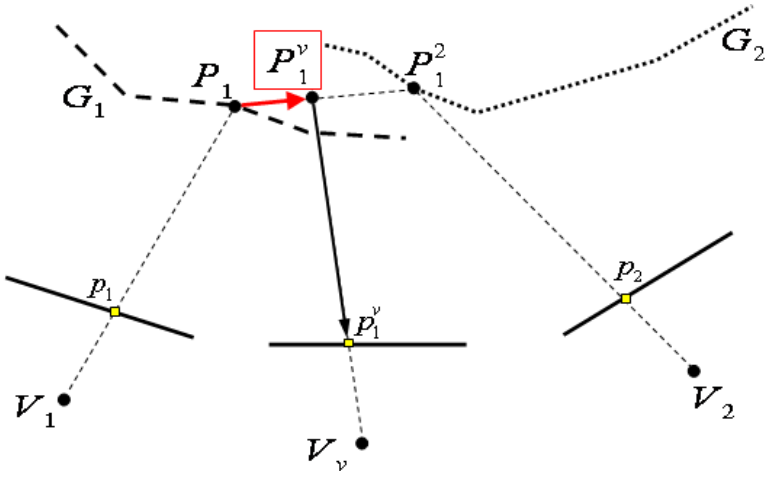


Figure 8.8: Virtual view synthesis with floating geometry. Vertex P_1 is floated to an intermediate position P_1^v between P_1 and P_1^2, then it is projected to the virtual view giving pixel p_1^v.

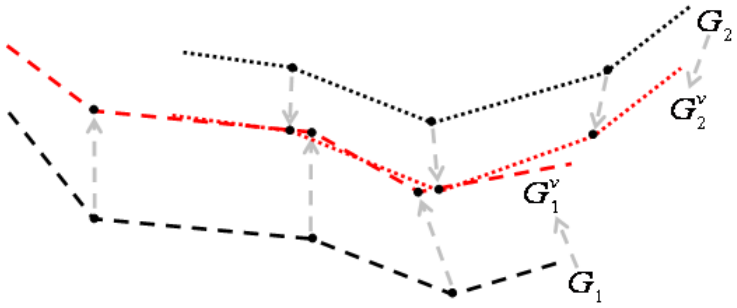


Figure 8.9: Virtual view synthesis with floating geometry. G_1 and G_2 are floated to G_1^v and G_2^v respectively. As a result, the geometries are consistent with each other, and texture misalignments are reduced.

fields of each views are not constrained to be coherent with each others, such that combining the flow fields do not inverse the process.

In the figure 8.9, the vertices from G_1^v and G_2^v are not perfectly overlapped because the two geometries have different sets of vertices. This illustrates that there is no matching between the vertices of the geometries because the motion estimation is performed between pixels in the original views. If two vertices have the same semantic meaning in two views, then only in this case they should overlap onto each other. This is the case for example if a single geometry is used.

8.4 Results on polygon soup

The floating geometry principle is now applied on the polygon soup representation introduced in this thesis. Two different scenarios are evaluated. First the floating geometry is computed at the acquisition side of the system, such that transmission issues are ignored. This scenario involves that all input views and their associated geometry are available, therefore the floating geometry is computed between multiple geometries. In this scenario, floating geometry will be first evaluated using one view. Then, floating geometry will be evaluated with two views and their associated polygon soup. This will show the improvement of consistency between multiple geometries. In the second scenario, floating geometry will be computed at the user side where only the polygon soup without redundancies is available. The advantage is that floating data has not to be transmitted since it is computed at the user side. In this case, the reduced polygon soup can be seen as a single geometry since redundancies have been reduced. This will show that the floating geometry is compatible with the polygon soup representation and that texture misalignments can be reduced with this method.

During all these experiments on floating geometry, the size of the quads used for view synthesis is reduced to 2×2 pixels such that the resolution of the polygon soup is increased and so is the precision of the floating geometry. This resolution during view synthesis must be differentiated with the one during data transmission. Indeed, the size of the quads is an important parameter for reducing the data before transmission by using larger quads. However, once transmitted, the quads can be split to very small size such that a dense polygon soup takes advantage of a dense motion estimation during floating geometry computation.

For motion estimation, the motion estimator of Urvoy et al. [UCP+09] is used. This motion estimator uses variable size block matching with some regularization.

8.4.1 Floating geometry at the acquisition side

One view projected into another original one. We first apply the floating geometry on one polygon soup extracted from one view only, and evaluate the texture misalignment when projecting this view into another original view. The color images and polygon soup associated to this other input view are used to compute the floating geometry. For evaluation, comparison of the synthesized view without and with floating geometry is performed. In addition, the PSNR between original and synthesized views

is computed. Since only one view is projected, disoccluded areas appear and these white areas are not included in the PSNR computation.

For this evaluation, V_3 is projected into V_1 using the polygon soup extracted from V_3 before redundancy reduction (i.e. full quadtree is used). We call this polygon soup Q_{V_3}. The floating geometry is performed such that Q_{V_3} is floated onto Q_{V_1} giving the floated polygon soup $Q_{V_3}^{V_1}$. The maximum size of the quads is 2×2 pixels.

Figure 8.10 shows the synthesized images I_3^1 without floating geometry and with floating geometry as well as the original image I_1. Images of the differences between original and synthesized images are also given. Without floating geometry, errors are particularly visible at texture edges, which corresponds to texture misalignments. With floating geometry, errors are reduced. However, small errors are still visible.

Concerning the objective quality measure, the PSNR has increased by 2.5 dB with the floating geometry (32.5 dB without vs 35.0 dB with).

This first experiment has shown that applying floating geometry to one polygon soup extracted from one view helps to reduce texture misalignments when it is projected into another view. Here, the side view was an original input one such that texture misalignments could be directly computed and compensated using the color image and geometry information of this input view.

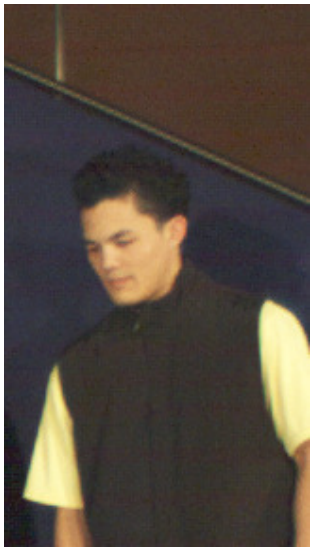
One view projected into a virtual view. The same experiment is now performed for a virtual view V_2 such that the floating geometry is now interpolated between the two positions given by the two side views V_1 and V_3. The original view V_3 is projected into virtual view V_2 using the polygon soup Q_{V_3}. Since $Q_{V_3}^{V_2}$ is not available, then the geometry is interpolated between Q_{V_3} and $Q_{V_3}^{V_1}$.

Figure 8.11 shows the synthesized images I_3^2 without floating geometry and with floating geometry as well as the original image I_2. Images of the differences between original and synthesized images are also given. As previously, the errors are smaller with the floating geometry than without. Note that the errors without the floating geometry are smaller than in the previous example (i.e. higher PSNR). This can be explained by the fact that the viewing angle between V_3 and V_2 is smaller than the one between V_3 and V_1, therefore texture misalignments are also smaller.

Concerning the objective quality measure, the PSNR has increased by 1.2 dB with the floating geometry (35.47 dB without vs 36.71 dB with). This confirms that the floating geometry helps to reduce texture misalignments even when synthesizing virtual views. Note that the PSNR values are both higher than in the previous example, this confirms that the texture misalignments are smaller when viewing angle is smaller. This also implies that the gain obtained with the floating geometry is smaller (1.2 db instead of 2.5 dB previously).

Two views projected into a virtual view. The two previous examples have shown that floating geometry helps reducing texture misalignments when using one view and its associated geometry. Since one view is not sufficient to prevent disocclusion areas, we now apply the floating geometry on the polygon soup representation extracted from two

(a) I_3^1 - Without floating geometry - PSNR = 32.5 dB



(b) I_1 Original view

(c) I_3^1 - With floating geometry - PSNR = 35.05 dB

(d) Difference without floating geometry

(e) Difference with floating geometry

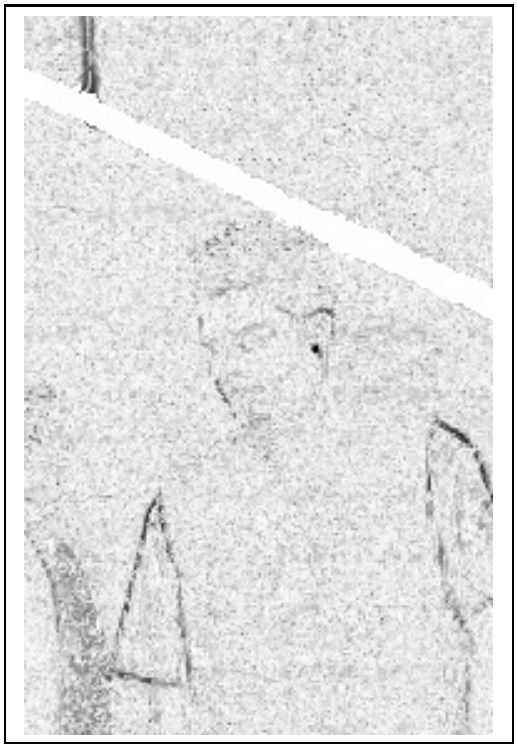
Figure 8.10: One view projected into another original one. Comparison without and with floating geometry.

(a) I_3^2 - without floating geometry - PSNR = 35.47 dB

(b) I_2 Original view

(c) I_3^2 - with floating geometry - PSNR = 36.71 dB

(d) Difference without floating geometry



(e) Difference with floating geometry

Figure 8.11: One view projected into a virtual view. Comparison without and with floating geometry.

views: the quadtree of each view is extracted and the floating geometry is computed for each quadtree. Then virtual view is synthesized using the synthesis methods described in chapter 6 so that no holes appear anymore and a complete view is synthesized.

In this experiment, virtual view V_2 is synthesized using original views V_1 and V_3 and their associated polygon soup Q_{V_1} and Q_{V_3}. Figure 8.12 shows the results of this experiment. The errors with floating geometry are smaller than without. On the contrary to previous examples, errors that appear in sub-figure (d) without floating geometry are splitted, with a small shift in between (e.g. on the left, the arm, the face and the white line in the wall are paired up). This double occurrence of errors corresponds to texture misalignments coming from the two views. Since geometries of both views are not consistent with each others, projected textures are not aligned in the synthesized view. This results in blur or splitting artifacts as can be seen in sub-figure (a) with the white line in the background or less clearly with the arm and face of the character. After applying floating geometry, sub-figure (e) shows that the splitting of the errors is reduced which proves that floating geometry improves the geometry consistency of multiple models by floating them one onto each other.

Concerning the objective quality measure, the PSNR values are 37.13 dB without floating geometry and 37.29 dB with floating geometry. These values are high because the two projected views are blended together giving an average error smaller than when one single view is used. A slight increase of the PSNR values is obtained with the floating geometry

Compared to the previous experiment where only one view was used, here the PSNR has increased by 0.58 dB (from 36.71 dB to 37.29 dB). However, overlapping two complete views like this creates strong ghosting artifacts around depth discontinuities as already shown in chapter 6 figure 6.15. These artifacts do not strongly affect PSNR values but strongly affect the subjective quality since they usually appear inside a uniform area. Moreover, the strong redundancies between the two views may represent a data overload considering real time view synthesis capabilities. Therefore, in the next experiment the redundancy reduction described in 5 is applied between the two views and the floating geometry is evaluated with this reduced polygon soup.

Two reduced views projected into a virtual view. The floating geometry method is now combined with redundancy reduction method so that the data load is reduced as well as ghosting artifacts around discontinuities. On one hand, redundancies between two views are reduced and a virtual view in between is synthesized. On the other hand, floating geometry is first computed between two views, then redundancies are reduced using the floating vertices computed and finally virtual view is synthesized and compared with the one synthesized without floating geometry.

Figure 8.13 shows the results. Virtual view V_2 has been synthesized using V_3 and V_1. Redundancies between Q_{V_3} and Q_{V_1} have been reduced with V_3 as the reference view. The images of differences in sub-figures (d) and (e) show that the errors have been reduced with the floating geometry.

The PSNR has increased by 1.0 dB with the floating geometry method (from 35.5 dB to 36.5 dB).

(a) Synthesis - without floating geometry - PSNR = 37.13 dB

(b) I_2 Original view

(c) Synthesis - with floating geometry - PSNR = 37.29 dB

(d) Difference without floating geometry

(e) Difference with floating geometry

Figure 8.12: Two views projected into a virtual view. Comparison without and with floating geometry.

This experiment has shown that floating geometry, when computed at the acquisition side of the system, helps reducing texture misalignments when synthesizing virtual views using a reduced polygon soup.

Summary Floating geometry has been performed at the acquisition side of the system. The goal was to evaluate the reduction of texture misalignments, step-by-step, when all information about original views are available. First, one view only was projected in original and virtual views. It has shown that texture misalignments are reduced and PSNR values increased in both original and virtual views. Then, floating geometry was evaluated using two views. If the views are full, then texture misalignments are slightly reduced but ghosting artifacts appear and redundancies are high. On the other hand, if the two views are reduced, as in our proposed representation, then ghosting artifacts are suppressed, redundancies are reduced, and floating geometry reduces texture misalignments with a PSNR increase of 1dB.

8.4.2 Floating geometry at the user side

A Different scenario. The previous experiments have been performed at the acquisition side such that full polygon soup and original images were available for each view (as for multiple geometries representation). The floating geometry was thus computed such that one geometry was floated onto another one. However, if the polygon soup has to be transmitted, then the extra information due to the floating geometry computation may be an obstacle to low bit rate transmission. We now consider the scenario where the reduced polygon soup (as it is described in chapter 5) is transmitted through a band limited channel and floating geometry is computed using this reduced polygon soup, at the user side of the system. The difference with the previous experiment is that the polygon soup is now a single geometry since redundancies have been reduced. Therefore, the back-projection step of the floating geometry computation is based on the single geometry itself instead of multiple geometries as in the previous experiment.

Three reduced views projected into a virtual view. In this experiment, original views V_1, V_3 and V_5 are used. The reduced polygon soup is first computed, and then given as an input to the floating geometry computation. Figure 8.14 shows the results of view synthesis with and without floating geometry. We can see that texture misalignments around edges have been reduced.

Tables 8.1 and 8.2 give the PSNR values obtained for virtual views V_2 and V_4 and sequences *Breakdancers* and *Ballet*. For *Breakdancers*, the PSNR measure has increased by 1.22 dB (from 35.29 dB to 36.51 dB) for virtual view V_2, and by 1.59 dB (from 34.65 dB to 36.24 dB) for virtual view V_4. Similarly, for *Ballet* sequence, the PSNR measure has increase by 1.29 dB (from 33.71 dB to 35.00 dB) for virtual view V_2, and by 1.53 dB (from 35.27 dB to 36.80 dB) for virtual view V_4.

Finally, figure 8.15 focuses on strong artifacts already mentioned in this chapter and chapter 6. Floating geometry succeeded to correct the texture misalignment in the character's face (sub-figure (a) and (b)). However, the white line in *Breakdancers* as

(a) Synthesis - without floating geometry - PSNR = 35.5 dB

(b) I_2 Original view

(c) Synthesis - with floating geometry - PSNR = 36.5 dB

(d) Difference without floating geometry

(e) Difference with floating geometry

Figure 8.13: Two reduced views projected into a virtual view. Comparison without and with floating geometry.

(a) Synthesis - without floating geometry - PSNR = 35.29 dB

(b) I_2 Original view

(c) Synthesis - with floating geometry - PSNR = 36.51 dB

(d) Difference without floating geometry

(e) Difference with floating geometry

Figure 8.14: Floating geometry computed at the user side. Three reduced views projected into a virtual view. Comparison without and with floating geometry.

Breakdancers	no floating geometry	floating geometry	gain
virtual view V_2	35.29 dB	36.51 dB	+1.22 dB
virtual view V_4	34.65 dB	36.24 dB	+1.59 dB

Table 8.1: Comparison of PSNR values without and with floating geometry using three views and reduced polygon soup.

Ballet	no floating geometry	floating geometry	gain
virtual view V_2	33.71 dB	35.00 dB	+1.29 dB
virtual view V_4	35.27 dB	36.80 dB	+1.53 dB

Table 8.2: Comparison of PSNR values without and with floating geometry using three views and reduced polygon soup.

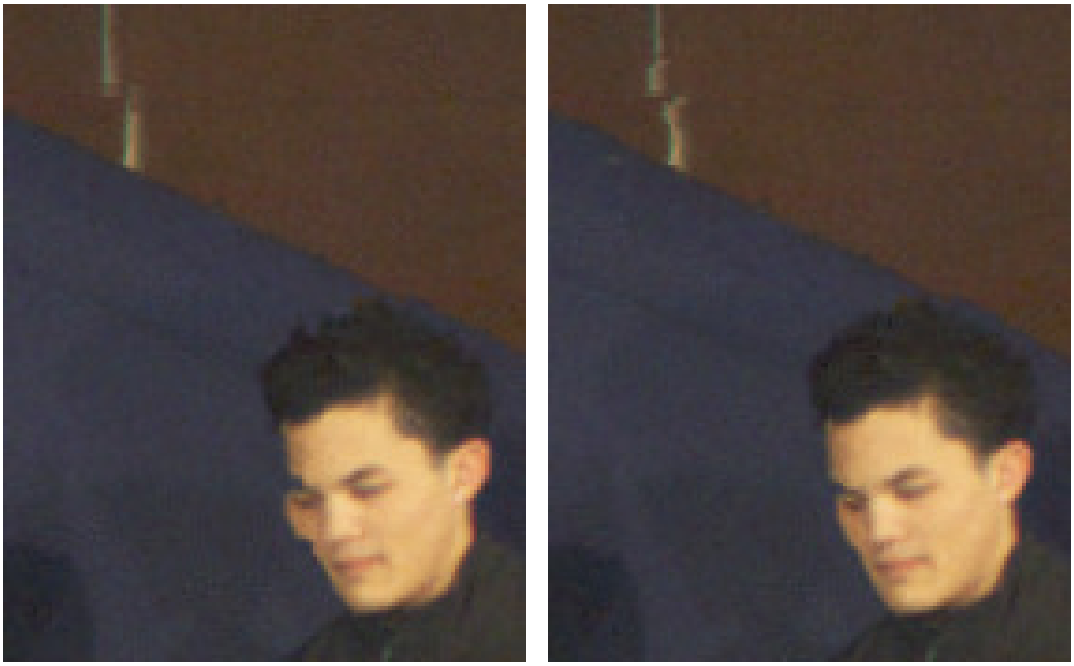
well as the stripes in *Ballet* could not correctly be aligned and artifacts are still visible. Two reasons may explain this limitation of the method.

- First, floating geometry is interpolated between the lateral views around the virtual one. However, if some vertices are occluded in one lateral view but disoccluded in the virtual view, then floating geometry cannot be computed and texture misalignments are not corrected. This occlusion issue also appears in the warped texture coordinate method but not the floating textures because computations are directly performed in the virtual view. One solution could be to extrapolate the floating geometry of non-occluded vertices to occluded vertices such that a correction is performed even if the geometry is occluded.

- Another reason comes from the limitation of the motion estimation. Indeed, if the texture deformation after projection is too strong, the motion estimation between projected texture and original one may fail. This effect increases as the viewing angle between views increases. For example, floating geometry computation is more difficult between views V_1 and V_5 than between views V_1 and V_3.

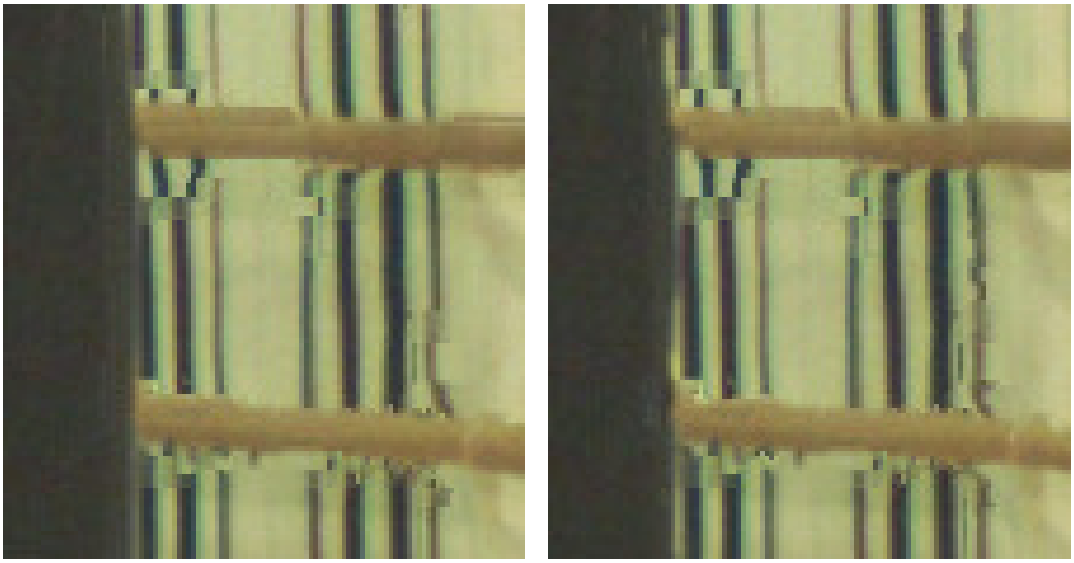
8.5 Conclusion

This chapter has introduced a new method for correcting texture misalignments that appear in synthesized views and geometry inconsistencies between multiple views. The method is called floating geometry and consists in deforming the geometric model onto a position guided by motion estimation between synthesized and original images. This view-dependent geometry is computed as a pre-process using input views, and it is interpolated when virtual views are synthesized.



(a) *Breakdancers* - without floating geometry

(b) *Breakdancers* - with floating geometry



(c) *Ballet* - without floating geometry

(d) *Ballet* - with floating geometry

Figure 8.15: Comparison of artifacts with and without floating geometry.

The method has been evaluated within two scenarios: one scenario where floating positions are computed at the acquisition side of the system and one scenario where they are computed at the user side of the system. In the first scenario, all original views and geometry are available, thus an accurate floating geometry can be computed. However, the data about floating geometry has then to be transmitted which increases the data load. In the second scenario, the floating geometry is not transmitted but computed at the user side, which saves data load but increases the complexity at the user side.

Results have shown that texture misalignments in virtual views can be reduced and that consistency between multiple geometric models is improved. Moreover, with the tested data set, results have shown that the floating geometry method is compatible with the polygon soup representation proposed in this thesis. An increase of up to 1.59 dB in PSNR measure is obtained when synthesizing virtual views at the user side of the system.

Results have also shown that the efficiency of the floating geometry depends on the viewing angle between input views. The efficiency decreases as the viewing angle increases. Moreover, occluded areas have raised an issue since their floating position cannot be computed although they may need correction when they are disoccluded in virtual views. An extrapolation of the floating geometry has been mentioned but not tested yet.

This floating geometry method may be compared to floating textures and warped texture coordinates methods [EDS+08]. The main difference is that geometry is floated instead of texture. For both methods, we have seen that perfect texture alignment is not guaranteed because the computed flow fields are not necessarily coherent with each others. More experiments and comparisons on the same multi-view video sequences with free viewpoint navigation could help identifying advantages and drawbacks of these methods.

The floating geometry method has also been designed with temporal aspects in mind. Indeed, as the geometry is floated in space between multiple views, it could also be floated in time. This method could help improving consistency in time and could also be used for time interpolation or time sub-sampling such that missing intermediate frames would be interpolated using floating geometry. Moreover, for compression considerations, the floating information of a polygon could be coded instead of coding a new quad at each time instant. The concept of *motion tubes* studied by Urvoy [UCP+09] could be extended to that purpose.

Chapter 9

Conclusions and perspectives

Contents

This thesis has presented a new representation for multi-view video. This representation is compact and takes into account in a unified manner the compression and view-synthesis stages of the system. In this chapter, the contributions of this study are summarized and possible improvements are presented. Finally, we propose some directions for future research.

9.1 Summary of contributions

9.1.1 Chap. 3: Study of existing representations

According to the targeted multi-view video system and associated constraints introduced in chapter 1, we have studied the properties of several existing representations and analysed the pros and cons each of them. This led us to few conclusions for the choice of a representation adapted to multi-view video:

- Geometry information is necessary for synthesizing virtual views if a sparse camera arrangement is used. Multi-view video plus depth (MVD) can be used as an input for constructing the geometry representation.

- Polygonal primitives have several advantages for compactness and view synthesis performances. Therefore a polygon-based representation may be well adapted to a multi-view video system.
- Using single or multiple textures/geometries influences the compromise between compactness and quality. The representation should contain both single and multiple textures/geometries, at a local level. In other words, some areas of the scene should be represented with multiple textures/geometries (for quality) and some areas should have only one occurrence (for compactness).
- Artifacts in virtual views are often due to inaccuracies of a single model or inconsistencies between multiple models. The representation should deal with this issues for good quality of virtual views.
- A standardized compression method exists for multi-view video. However, it is not adapted to the properties of depth maps and no other compression method has yet been adopted for geometry in the context of multi-view video with arbitrary scenes. Particularly, depth discontinuities must be preserved and require special attention within the compression step. Therefore, the representation should be compatible with an efficient compression method that avoids visual artifacts in virtual views.

9.1.2 Chap. 4: Overview of the representation

This chapter has introduced our proposed representation and its properties. It is made of a set of 3D polygons not necessarily connected to each others and possibly overlapping. The polygons are stored as 2D quads with depth values at each corners. They are extracted from the input depth maps using a quadtree decomposition technique. The polygons can evolve through space and time. The original color images are used to texture each polygon.

These choices aim at providing a compact representation which can be efficiently compressed and allows good quality virtual views with reduced view synthesis complexity. The rest of the thesis was dedicated to the construction and validation of this representation for a multi-view video system.

9.1.3 Chap. 5: Construction of the polygon soup

This chapter has presented the construction of the polygon soup. First, a set of quads is extracted using a quadtree decomposition of the depth maps. This step preserves depth discontinuities and geometric details while keeping as few quads as possible. To do so, an error measure based on the re-projection shift has been proposed. Results have shown that it allows to better control the compromise between compactness and geometry approximation. Second, an original method for the reduction of inter-view redundancies has been presented. It uses a priority order to select reliable quads while keeping as few quads as possible. Results have shown that about 65% of quads have been removed compared to the full polygon soup.

A possible improvement of the construction would be to use an image-based error measure so that the quality of synthesized views is taken into account for tuning the parameters. However, to prevent excessive subdivisions of the quads, such a measure should be adapted to the properties of synthesized views and to the relations between texture and geometry errors.

9.1.4 Chap. 6: Virtual view synthesis

This chapter has presented the view synthesis stage based on the polygon soup representation. Most of the techniques presented in this chapter are known and already used for other applications or other representations. Therefore, the main contribution of this chapter is the adaptation of these techniques to the polygon soup representation, and the evaluation of the quality of the virtual views. First a method to remove cracks between polygons of different size has been applied. Second, the blending of the textures in overlapping areas has been detailed. It provides smoother transitions between polygons. Finally, classical image processing algorithms have been performed to enhance the virtual views. It includes inpainting of unknown areas and edge filtering of object boundaries.

The evaluation of the virtual views has shown that good quality images were obtained, at a PSNR of about 34-35 dB, and without ghosting artifacts. However, some artifacts appear: cut of line, blocky artifact, distortions. It is mainly due to geometry inaccuracies or inconsistencies of polygons extracted from different views. A possible solution to reduce these artifacts is studied in chapter 8.

The different processes have not been implemented yet for real-time performances, therefore no quantitative results are available concerning the computational complexity. The proposed view-synthesis method includes quadtree subdivision, projection and texture mapping, blending, inpainting and edge filtering. Compared to view-synthesis methods using depth image-based representation like MVD, a smaller set of polygons replace the points, no median filtering that removes sampling artifacts is needed, neither adaptive processing of edges that removes ghosting artifacts.

9.1.5 Chap. 7: Compression of the polygon soup

In this chapter, an adapted compression technique of the polygon soup has been proposed. The method takes advantage of the quadtree structure that helps to efficiently retrieve the position and size of each quad. The compression method consists in predicting the depth values of the quads' corners using the already coded neighbor quads. Then, prediction residues are quantized and coded with context adaptive binary arithmetic coding (CABAC).

The performances of the compression method have been compared with an existing approach that uses multi-view plus depth representation (MVD) compressed with multi-view coding technique (MVC). Results have shown that the proposed method gives a slightly higher quality of synthesized views (+0.3 dB at most) at medium and high bit rates. Moreover, no ghosting artifacts were observed with the compressed polygon

soup representation, whereas these artifacts appear when using the compressed MVD representation.

The MPEG's 3DV group is currently studying how to represent and compress the depth information of MVD data. The proposed polygon soup and associated compression technique could be considered as an alternative to these methods. However, the compression method is not yet mature, and more improvements are necessary to reduce distortions: first, the method is not efficient at low bit rates. An optimization of the parameters of the quadtree could help improving the performances at such bit rate. The thresholds that control depth discontinuities and geometry accuracy can be tuned for this purpose. Second, the object boundaries exhibit small distortions due to the compression step. An adaptation of the compression method is necessary. A possible improvement could be to modify the parsing of the quads during the prediction step. Indeed parsing the quads from the bigger size to the smallest size could help predicting smallest quads around discontinuities at last, thus prediction from connected neighbors quads and not only from top left neighbors quads would be possible.

9.1.6 Chap. 8: Floating geometry

This chapter has introduced a dynamic representation that helps reducing texture misalignments that appear in synthesized views. The method is called "floating geometry" and consists in deforming the geometric model onto a position guided by motion estimation between synthesized and original images. This view-dependent geometry is computed as a pre-process using input views, and it is interpolated when virtual views are synthesized. This method is applied to the polygon soup and may be applied to any other geometry.

Results have shown that texture misalignments in virtual views can be reduced and that consistency between multiple geometric models is improved. An increase of up to 1.59 dB in PSNR measure is obtained. Results have also shown that the efficiency of the floating geometry depends on the distance (viewing angle) between input views. The efficiency decreases as the viewing angle increases. Moreover, occluded areas have raised an issue since their floating position cannot be computed although they may need correction when they are disoccluded in virtual views. An extrapolation of the floating geometry has been mentioned but not tested yet.

9.2 Perspectives

We have already mentioned that the construction of the polygon soup could be tuned using an image-based error measure, such that photo-consistency is checked across the views. However, this might result in high subdivision of the quadtree even with only small geometry errors. Indeed, image-based error measures are not adapted to errors due to geometry approximation and re-projection. In addition, we have seen that floating geometry helps reducing texture misalignments due to geometry errors, but still strong distortions could not be corrected. Therefore, the result of the floating geometry could be used as a feedback loop for the construction of the polygon

soup: if strong image errors still appear after the floating geometry, then the original quads corresponding to the distorted area should be added to the polygon soup. Thus, redundancies would be added at a local level, and it would allow to better tune the compactness and image quality trade-off.

Moreover, we have seen that the quality of virtual views decreases as the distance of projection increases. For large navigation range, the central view which is used as reference view may have insufficient quality when projected to a large distance. In this case, the combination of two or more reference views could help preserving a good navigation range and image quality. This would be similar in spirit with the Depth Enhanced Stereo (DES) format [SMM+09] (mentioned in section 3.3) that uses two reference views, each associated with LDV representation.

The consideration of the temporal dimension is still to be investigated for our compression method. Similarly to the video compression approach in Urvoy's work [UCP+09], the representation is defined as a set of polygons. The extension to the temporal dimension could consider the temporal evolution of a polygon. A frame would then be reconstructed by sets of polygons coming from different viewpoints (as already seen) but also from different temporal time instant. This notion of evolution of the polygons leads to a dynamic version of the polygon soup.

The floating geometry method has also been designed with temporal aspects in mind. Indeed, as the geometry is floated in space between multiple views, it could also be floated in time. This method could help improving consistency in time and could also be used for time interpolation or time sub-sampling such that missing intermediate frames would be interpolated using floating geometry. Moreover, for compression considerations, the floating information of a polygon could be coded instead of coding a new quad at each time instant.

Appendix A

Fusion of background quads

Contents

This annex presents an additional method for reducing the number of quads in the polygon soup. Since only the number of quads has been evaluated and not yet the compression and view synthesis stages, it is presented in this annex rather than in the chapters of this study.

A.1 Introduction

During the construction of the polygon soup, the input depth maps are decomposed into a quadtree structure. Three criteria are respected during this decomposition: discontinuity preservation, geometry preservation and maximum quad's size. The discontinuity preservation is achieved by subdividing the quads until no one connects both foreground and background areas. This subdivision results in a high number of small quads around all depth discontinuities. Figure A.1 illustrates this process: a depth discontinuity separates background quads and foreground quads. Many small quads are created around the depth discontinuity. This can be seen also in figure A.2 that shows the final polygon soup extracted from *Breakdancers* sequence. It is shown from a lateral viewpoint such that decomposition of the background and foreground areas can be observed.

Although this decomposition method preserves very well depth discontinuities, the geometry around may be very simple and planar, such that it could be approximated by big quads. Therefore, using so many small quads is inefficient for the compactness of the representation.

In order to reduce the number of quads in the polygon soup, we propose to fuse the background quads that were subdivided because of a depth discontinuity. We will

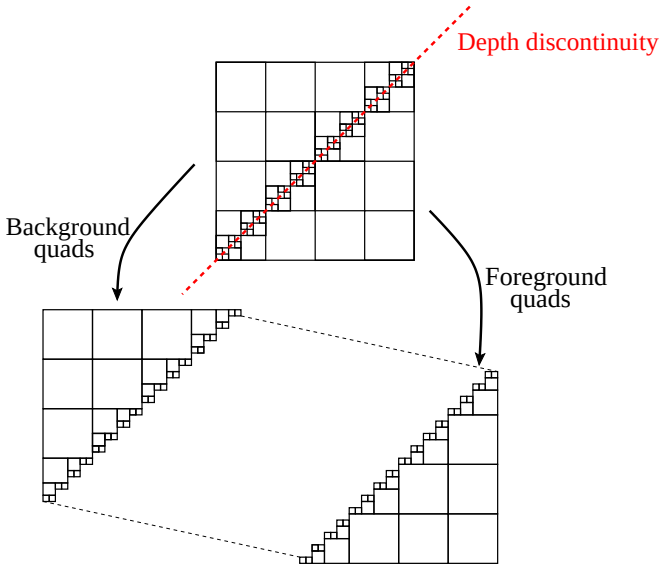


Figure A.1: Around depth discontinuities, quads are subdivided resulting in background quads and foreground quads.

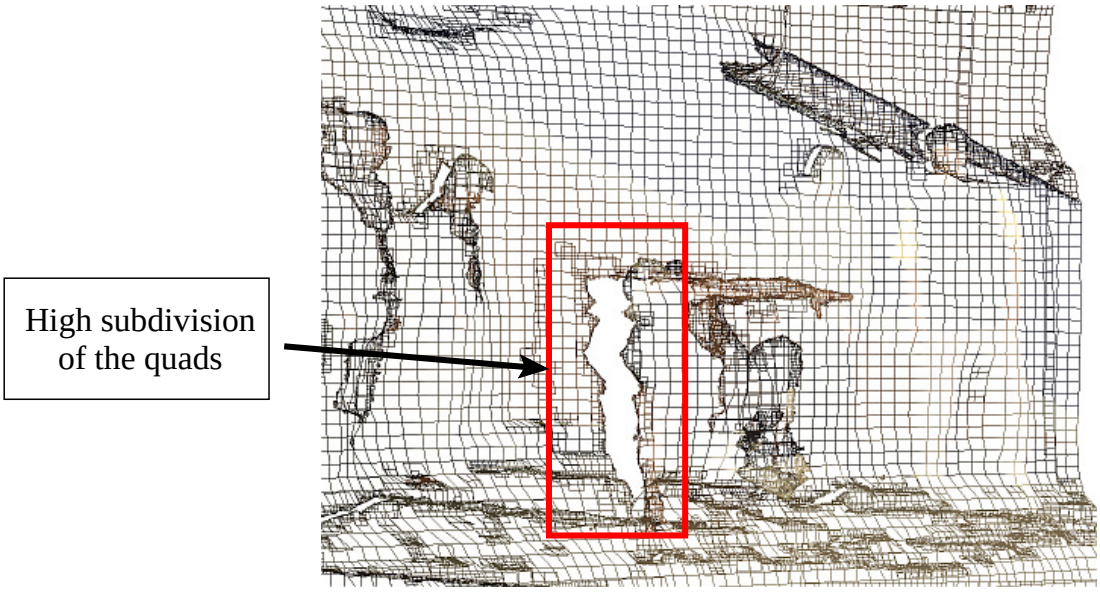


Figure A.2: High subdivision of the quads situated around depth discontinuities.

see in the following that our proposed method can fuse either background quads or foreground quads but not both. We have decided to fuse background quads since this area of the scene is often more smooth and planar than the foreground. To fuse the quads, few issues have to be solved concerning the depth values and textures of the new quads, and the modification of the quadtree structure. A method that deals with these issues is now detailed.

A.2 Fusion of the background quads

The goal is to fuse small quads in the background in order to get bigger quads. This process is performed on each quadtree obtained after the construction of the polygon soup, explained in chapter 5. It results in a modified quadtree with fewer and bigger quads. In the following, background is denoted *Bg* and foreground *Fg*. Considering the quadtree structure, the fusion could be pretty simple: four leaves of the tree just have to be combined into their father node. However, since the quads are situated around discontinuities, some of the leaves belong to the Fg and must not be fused with the Bg. This involves to slightly modify the quadtree structure such that fused Bg quads can also have children Fg quads. Figure A.3 illustrates this fusion and modification of the quadtree structure. In order to signal a fused node in the quadtree structure, a 1-bit flag is used. If '1', it means that the node contains a quad and also children quads.

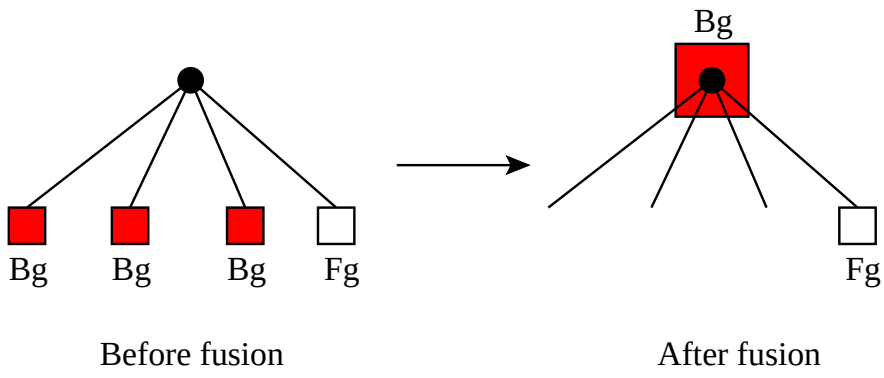


Figure A.3: Fusion of background quads and modification of the quadtree structure. Before the fusion, many quads are situated in the Bg but there are also Fg quads that cannot be fused. After the fusion, the father node contains the fused Bg quad and has also the child Fg quad. It is a fused node.

Once the quadtree structure is modified, we can explain the fusion process. The main issue here is to define new corners and transparent texture in the occluded areas. Indeed, since the fused quad contains also one or more Fg children, it is occluded by them. Therefore, the occluded corners of the fused Bg quad have to be estimated, and the occluded area must have transparent texture because it is unknown. Figure A.4 illustrates this problem.

The fusion process consists in parsing the quadtree from bottom to top. We start

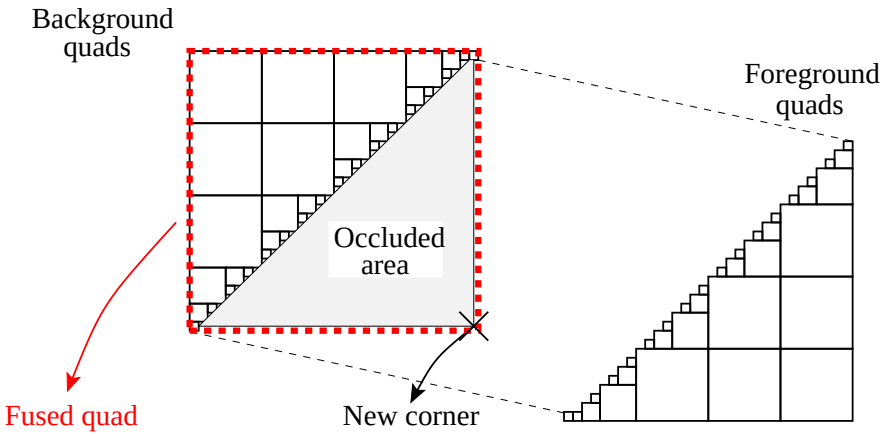


Figure A.4: Creation of the fused quad: occluded areas must have transparent texture, and the depth of the new corners must be estimated .

from the leaves and try to fuse children quads while climbing up the tree. Therefore, one fusion involves maximum four children quads at a time, in the previous figure A.4 the result of all the fusions is shown, not the intermediate fusions.

Each quad has four children quads (except leaves). If there is a discontinuity inside the quad, it means that some children quads are Bg and some are Fg. Thus we want to fuse the Bg children quads. To do so, non-occluded corners of the fused quad are associated with those of its Bg children. Then the computation of the new corners occluded by Fg children quads is simply performed by interpolating non occluded Bg children as if the surface was flat. This results in the fused quad. Before creating this new quad, the geometry preservation criterion has to be checked, which is performed with the same method explained in section 5.1. If the criterion is not respected, then the quads are not fused. Finally, the transparent area is managed by an additional texture image dedicated to the fused quads: it contains transparency in areas delimited by the Fg children quads. Note that this second texture doesn't have to be transmitted since it can be created at the user side using the positions of Fg children quads. On the other hand, if Fg children quads where also fused, then the transparent area could not be recovered anymore and additional texture images would have to be transmitted. This is the reason why only background quads are fused. All this process in summarized in the algorithm 2 below.

A.3 Results

The aim of the fusion process was to reduce the number of quads in the polygon soup while preserving the geometry accuracy. Therefore, we now evaluate the performance of this method in terms of the reduction of number of quads. The small modification of the quadtree structure requires adapting the compression and view synthesis steps, which has not been done yet. Therefore, no rate/distortion nor PSNR results are available

Objective: Parse a quadtree (bottom-up) and fuse Bg quads when possible.

```
1 foreach quad do
2     if DiscontinuityInside (quad) then
3         identify Bg and Fg children quads;
4         Compute new corner(s) and fuse Bg children quads ;
5         if PreservesGeometry (fusedQuad) then
6             Insert fused quad in quadtree;
7             Add transparency to occluded area of fused quad;
8             Remove Bg children quads from quadtree;
          end
      end
  end
```

Algorithm 2: Fusion of background quads

for the moment.

Table A.1 compares the number of quads before and after the fusion of background quads. A reduction of 26% of quads is achieved for sequence *Breakdancers* and 20% for sequence Ballet.

	Breakdancers	*Ballet*
Before fusion	23558	27615
After fusion	17334	22186
Quad reduction	-26%	-20%

Table A.1: Number of quads before and after the fusion process.

Finally, figure A.5 gives a zoomed area of the polygon soup before and after the fusion. The polygon images show that many small quads have been replaced by bigger ones and thus the polygon soup is more compact. Finally the color images show that the image quality is preserved and the transparency is actually applied to the fused quads. Note that some small cracks appear in the images. Indeed the cracks elimination process presented in section 6.1 is not implemented in this example.

A.4 Conclusion

This annex has presented a new method for reducing the number of quads in the polygon soup representation. Starting from each reduced quadtrees obtained after the construction of the polygon soup, the idea is to fuse background quads that were subdivided because of depth discontinuities. To do so, each quadtree is parsed from

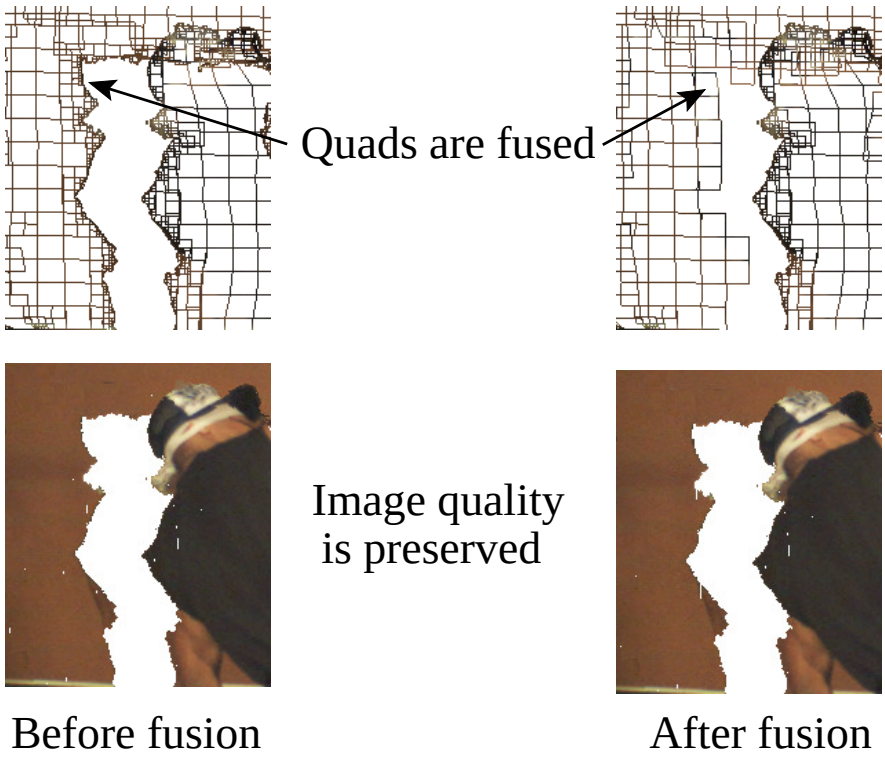


Figure A.5: Comparison of the polygon soup before and after the fusion of background quads.

bottom to top and background quads are fused into a bigger quad while foreground quads are preserved thanks to a small modification of the quadtree structure.

Results have shown that the number of quads could be reduced from about 20 to 25%. Thus it efficiently reduces the number of quads.

This method has not been fully evaluated. Indeed the view synthesis and compression steps have to be adapted to the new structure of the quadtree. We can expect that a lower bit rate would be obtained with this technique, but this has to be verified since the new quadtree structure requires an additional 1-bit flag at each node of the tree.

Finally, the fused quads can be seen as small billboards since their surface is bigger than the part of the scene they describe. In this sense, it can be compared to the impostor-based representations presented in chapter 3. More particularly, both techniques need to define the transparent surface of the polygon, which was also referred to as *texture clipping* in the chapter. However, the difference is that texture clipping had to be transmitted as a mask or computed with alpha matting or with additional depth maps, which either increase the bit rate or the view synthesis complexity of the system. In our proposed method, no such drawbacks appear since the transparent areas are recovered by simply reading the quadtree structure.

Appendix B

Représentation par soupe de polygones déformables pour la vidéo 3D

Contents

B.1 Introduction

L'année 2010 a été marquée par l'arrivée de la vidéo 3D pour le grand public. Le film 3D 'Avatar' au cinéma en décembre 2009 a rencontré un énorme succès[1], et les compagnies d'électronique annoncent la démocratisation des téléviseurs '3D-ready'[2]. En effet, les technologies ont suffisamment évolué pour pouvoir capter deux vues au même instant, les traiter, les transmettre et les afficher avec une bonne qualité. Ici, le terme '3D' est justifié par la visualisation stéréoscopique: la vision binoculaire humaine est exploitée pour obtenir une meilleure perception de la profondeur, donnant ainsi une sensation de relief. En fait, il n'y a qu'une seule image en plus comparé à la vidéo 2D classique,

[1] http://avatarblog.typepad.com/avatar-blog/2010/01/avatar-biggest-movie-of-alltime.html
[2] http://www.3dtvsource.com/sony-2010-3d-tv-lineup/

mais c'est suffisant pour montrer le potentiel de la vidéo 3D à améliorer la description d'une scène et la sensation de profondeur pour les utilisateurs.

De nombreux travaux de recherche sont à présent dirigés vers la **video multi-vue** afin d'augmenter le nombre d'images d'une scène au même instant [SKS05, Mag05, KSM+07, DKM+10]. La vidéo multi-vue apporte principalement deux fonctionnalités aux utilisateurs. La première est la navigation: l'utilisateur peut changer de point de vue dans la scène. La seconde est la visualisation auto-stéréoscopique: l'utilisateur profite de la stéréoscopie sans avoir besoin de lunettes spéciales.

La mise en place d'un système de vidéo multi-vue fait apparaître de nombreux défis à toutes les étapes de la chaîne: acquisition, représentation, transmission, synthèse de vue et affichage. L'objectif de cette thèse est de s'intéresser à l'étape de représentation. Cette étape est très importante car elle influence la quantité de données devant être transmise, la méthode de compression à utiliser ainsi que la complexité de calcul et la qualité finale de l'image. Les représentations de données existantes contiennent souvent des informations sur la géométrie de la scène en plus des images couleurs. Ces représentations ont leurs avantages et inconvénients. Nous souhaitons les analyser pour ensuite proposer une nouvelle représentation qui prennent en compte les problèmes de quantité de données, de méthode de compression, de complexité de calcul et de qualité d'image.

B.2 Représentations existantes

Dans le contexte de la vidéo multi-vue, une représentation est une description de la scène en utilisant un certain type de données. Le choix d'une représentation est important car elle influence les étapes du système: tout d'abord la représentation est construite à partir des données acquises, puis elle est compressée et transmise. Ensuite, du côté utilisateur, la représentation est décompressée et des vues sont synthétisées pour être affichées.

Dans le but de comparer les représentations entre elles, cinq propriétés ont été analysées: la complexité de construction, la compacité, la compatibilité avec la compression, la complexité de synthèse de vue, la qualité d'image et l'étendue de navigation possible. De plus, pour faciliter la comparaison entre représentations, le type de système visé et les contraintes associées à ce système ont été définis.

Quatre familles de représentations ont été étudiées, certaines contenant des sous-familles, donnant au final sept différentes représentations:

- Représentation basée images. Cette représentation n'utilise pas d'information géométrique, mais uniquement des images capturées par plusieurs caméras. C'est un avantage pour la compacité et la compression de cette représentation. En revanche, la qualité des images synthétisées et la navigation autour des images originales sont très réduites à cause du manque d'information géométrique.

- Représentation basée images plus profondeur. Elle utilise, en plus des images originales, des cartes de profondeur qui associent une valeur de profondeur à chaque

pixels de l'image. Deux types de représentations basées image plus profondeur ont été étudiés: multi-vues plus profondeur (MVD: multi-view plus depth) et profondeur par couche (LDV: layered depth video). La représentation MVD est illustrée dans la figure B.1. Elle a l'avantage de donner une bonne liberté de navigation avec des images de grande qualité. Mais la quantité de données est élevée. A l'inverse, la représentation LDV est compacte mais la navigation et la qualité d'image sont plus restreintes. Avec ces deux représentations, de nombreux travaux sont actuellement dirigés vers la compression et la synthèse de vue. La méthode de compression doit préserver le plus possible les contours de profondeur. La synthèse de vue fait appel à de nombreux post-traitements qui augmentent la complexité du côté utilisateur.

- Représentation basée surface. Elle modélise la surface d'une scène en utilisant un certain type de primitive géométrique. Les primitives les plus utilisées sont les polygones (connectés entre eux pour former un maillage) et les points. Les maillages de polygones sont très utilisés dans la communauté graphique pour représenter des modèles synthétiques. Par conséquent les cartes graphiques sont très performantes pour afficher de tels modèles. De plus, des polygones de grande taille peuvent être utilisés pour représenter des surfaces planes, ce qui permet d'augmenter la compacité. La compression des maillages de polygones représentants des objets seuls et souvent synthétiques aboutit à des standards de compression performants. En revanche, il semble que la compression de scènes arbitraires contenant des discontinuités de profondeurs soit moins performante et peu de contributions ont montré leur efficacité pour la vidéo 3D. En ce qui concerne la représentation utilisant des points comme primitive géométrique, elle est principalement utilisée pour modéliser des scènes complexes avec beaucoup de détails. En effet, dans ce cas il est plus efficace d'afficher un nuage de points plutôt qu'un maillage très fin de polygones dont la taille dans l'image peut être plus petite qu'un pixel. La représentation par points contient donc généralement une énorme quantité de points, ce qui est un inconvénient pour la compacité de la représentation.

- Représentation basée imposteurs. Elle utilise des plans 3D sur lesquels la texture des images est plaquée. Ils sont souvent employés pour modéliser des objets complexes: à la place de l'objet complexe est affichée son image, un imposteur. Deux types d'imposteurs utilisés pour des scènes réelles ont été étudiés: les billboards (litt. "panneau d'affichage") et les microfacettes. Les billboards sont des grands plans 3D qui permettent une extrême simplification de la scène. A l'inverse, les microfacettes sont des plans 3D plus petits et toujours orientés face à la caméra. Ces deux représentations sont très compactes mais aboutissent soit à une faible navigation et qualité d'image (billboards) soit à une complexité de synthèse de vues élevée (microfacettes).

L'étude des avantages et inconvénients des représentations existantes a permi de lister quelques conclusions concernant le choix d'une représentation pour la vidéo 3D:

Figure B.1: Exemple de vidéo multi-vues plus profondeur (MVD).

- L'utilisation d'informations géométriques est nécessaire pour synthétiser des vues de bonne qualité en utilisant des caméras distantes les unes des autres. Les cartes de profondeur peuvent êtres utilisées comme données d'entrées pour cette géométrie.
- Les primitives polygonales sont avantageuses pour la compacité et la synthèse de vues. Une représentation basée sur des polygones pourrait être bien adaptée à la vidéo 3D.
- Utiliser une ou plusieurs textures/géométries influence le compromis entre compacité et qualité. Par conséquent, une représentation doit pouvoir régler ce compromis en conservant plusieurs occurrences pour certaines parties d'une scène (pour la qualité) et en supprimant d'autres parties d'une scène (pour la compacité).
- Les artefacts qui apparaissent dans les vues virtuelles sont souvent dus aux imprécisions de la géométrie ou bien aux inconsistances entre plusieurs géométries. Ces défauts doivent être pris en compte dans la représentation.
- La représentation doit être compatible avec une méthode de compression efficace et qui ne crée pas d'artefacts visuel gênants.

B.3 Une nouvelle représentation

Cette section décrit la nouvelle représentation proposée dans cette thèse, appelée *soupe de polygones déformables* et illustrée en figure B.2. Les données d'entrées utilisées pour construire la soupe de polygones sont des données multi-vues plus profondeurs (MVD, figure B.1). Tout d'abord, les primitives géométriques utilisées sont des polygones 3D. Ces polygones ne sont pas nécessairement connectés entre eux et peuvent se superposer. Ensuite, ces polygones sont définis en 2D avec des valeurs de profondeur à chaque coin.

Ces polygones 2D sont extraits des cartes de profondeur grâce à une décomposition en quadtree, on les nomme *quads*. Pour une meilleure compacité de la représentation, les redondances inter-vues sont réduites de façon adaptative en supprimant certains quads de la représentation. Cette soupe de polygones peut être compressée soit par une méthode basée quadtree, soit par une méthode basée bloc. Concernant les textures, les images originales acquises par les caméras sont utilisées et plaquées sur les polygones 3D. Ces images sont compressées avec les méthodes existantes de compression vidéo multi-vues. Pour les vues synthétisées, des post-traitements sont ajoutés pour améliorer la qualité finale de l'image. Enfin, nous introduisons la notion de géométrie déformable en fonction du point de vue afin de réduire les artefacts et désalignements de textures dans les images synthétisées. Cette déformation peut aussi être appliquée dans le temps afin de considérer l'évolution temporelle d'un polygone.

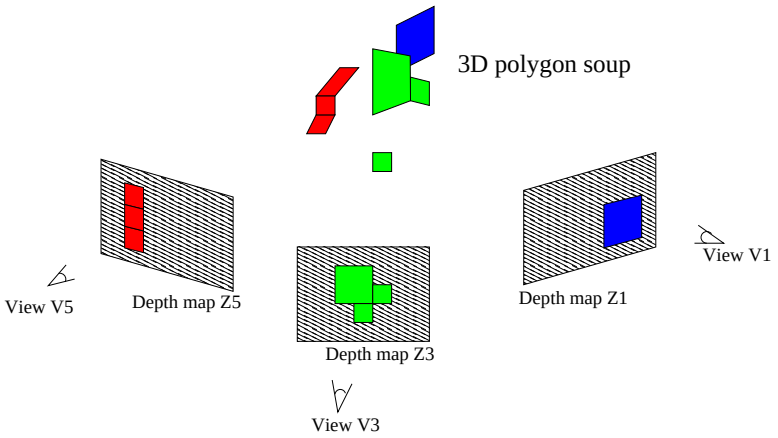


Figure B.2: Soupe de polygones. Chaque polygone 3D est définit par un polygone 2D (quad) et par des valeurs de profondeurs à chacun de ses coins.

B.4 Construction de la soupe de polygones

La soupe de polygones est construite à partir des cartes de profondeur. Deux étapes peuvent être distinguées: la décomposition en quadtree qui extrait un ensemble de quads pour chaque vue, et la réduction des redondances inter-vues qui supprime les quads redondants et peu fiables par ordre de priorité. Ces étapes sont illustrées dans la figure B.3.

Pour éviter l'apparition d'artefacts dans les vues synthétisées, la décomposition des cartes de profondeur en quadtree doit respecter les discontinuités de profondeur ainsi que les détails de la géométrie de la scène. Le fait de respecter les discontinuités de profondeur permet de ne pas utiliser un quad qui relierait à la fois un objet en avant-plan et un objet en arrière-plan. Le fait de respecter la géométrie de la scène permet de conserver les détails géométriques tout en utilisant des grands quads dans les zones planes. Dans ce but, une mesure d'erreur basée sur le décalage de re-projection est

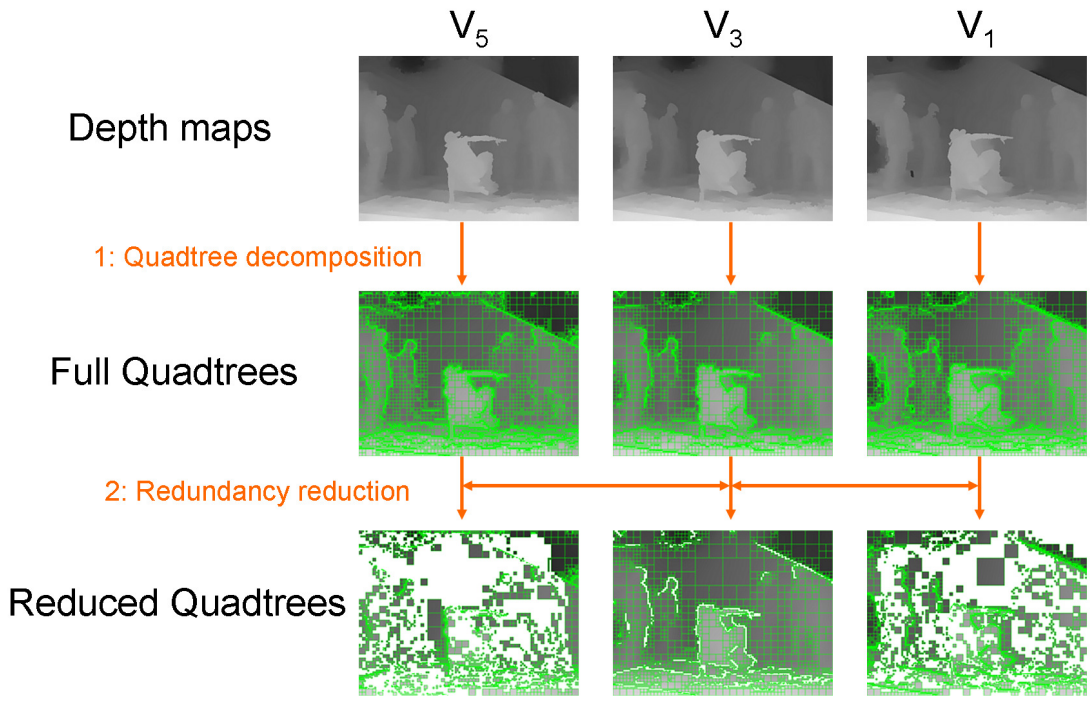


Figure B.3: Vue d'ensemble de la méthode de construction de la soupe de polygones.

proposée. Ce critère permet de prendre en compte l'étendue de la navigation et donc la qualité des vues virtuelles synthétisées. Ainsi, une vue centrale et une vue latérale n'ont pas la même finesse de décomposition car elles n'ont pas les même re-projections.

La décomposition en quadtree donne une représentation redondante car plusieurs polygones venant de vues différentes peuvent représenter la même zone de la scène. Par conséquent, la deuxième étape de la construction de la représentation est de réduire ces redondances. L'idée est de sélectionner les quads de façon itérative selon un ordre de priorité permettant d'ajouter d'abord les quads fiables et de grandes tailles. L'ordre de priorité dépend à la fois de la position des vues, de la taille des quads et de leurs profondeurs. Une itération consiste à prédire une vue avec l'ensemble des quads déjà sélectionnés et ajouter les quads manquants en fonction de l'ordre de priorité défini. En plus de réduire les redondances, cette étape permet aussi de supprimer les quads peu fiables qui pourraient créer des artefacts gênants dans les vues virtuelles.

Une fois la soupe de polygones construite tous les polygones sont utilisés pour synthétiser des vues. La figure B.4 illustre la soupe de polygones non-texturée projetée dans une vue virtuelle en dehors de la fenêtre de navigation afin de bien visualiser les discontinuités de profondeurs. La compacité de cette représentation est évaluée par le nombre de quads obtenus. Les résultats évalués sur deux séquences MVD acquises avec le même arrangement de caméras ont montré que la soupe de polygones contient environ 25 000 quads, ce qui est équivalent au nombre de quads pour une vue non réduite, et que environ 65% de quads ont été supprimés grâce à l'étape de réduction des redondances. Dans la suite, la soupe de polygones est évaluée aux étapes de synthèse de vues et compression.

B.5 Synthèse de vues virtuelles

Afin d'évaluer la qualité des images synthétisées en utilisant la soupe de polygones, des méthodes classiques de projection et de pré- et post-traitements ont été adaptées à la soupe de polygones. La synthèse de vues virtuelles est effectuée par projection des polygones en utilisant une carte graphique et la librairie openGL. La texture des images est plaquée sur chaque polygone correspondant. Avant cette étape de projection, une redivision des quadtree est effectuée afin d'éviter les artefacts de fissures entre polygones de tailles différentes. Après l'étape de projection, il est possible que des polygones issus de vues différentes se superposent dans la vue virtuelle. Afin d'adoucir les transitions entre polygones qui se superposent (dues à des inconsistances de couleurs ou de géométries entre vues), les couleurs de ces polygones projetés dans la vue virtuelle sont moyennées de façon adaptative avec une pondération dépendant de la position des vues originales en fonction de la vue virtuelle. Enfin, une dernière étape améliore l'image finale. Tout d'abord une méthode d'inpainting est appliquée, elle consiste à remplir les zones restantes qui ne sont pas prédites par la projection des vues (zones occultées dans les vues originales). Ensuite, un filtre passe-bas est appliqué sur les contours des discontinuités de profondeurs afin de donner un aspect plus naturel à ces contours qui sont en général trop nets après l'étape de projection. Ces différentes étapes

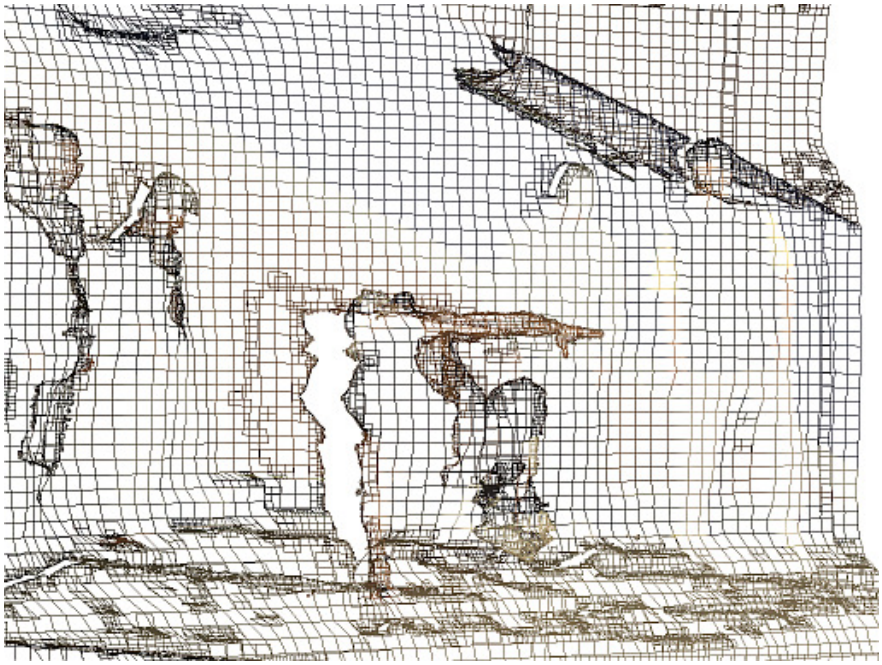


Figure B.4: Soupe de polygones obtenue après les deux étapes de la construction et projetée dans une vue virtuelle hors de la fenêtre de navigation.

de traitements sont résumées dans le schéma de la figure B.5.

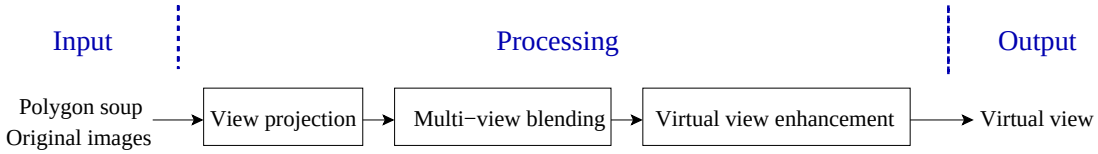


Figure B.5: Vue d'ensemble des étapes de la synthèse de vues virtuelles.

Il est à noter qu'aucun traitement spécifique n'est ajouté pour supprimer les artefacts appelés *artefacts fantômes* (ghosting artifacts en anglais). En effet, lors de la construction de la représentation, l'ordre de sélection des polygones est fait pour supprimer les petits polygones proches des discontinuités et qui sont généralement la cause de ces artefacts. La figure B.6 illustre l'avantage de supprimer ces petits polygones.

Pour évaluer la qualité des images synthétisées, les points de vues qui ont été synthétisés correspondent à des points de vues de caméras originales. Ainsi, une mesure d'erreur objective peut être utilisée pour comparer une vue originale avec celle synthétisée. La mesure d'erreur utilisée est le PSNR (*Peak Signal to Noise Ratio*) qui donne des résultats en décibels (dB). Des valeurs typiques de PSNR pour des images de bonnes qualités commencent entre 30 et 40 dB. Les résultats de deux séquences vidéos sur 25 instants temporels donnent une moyenne d'environ 34 dB. Les images

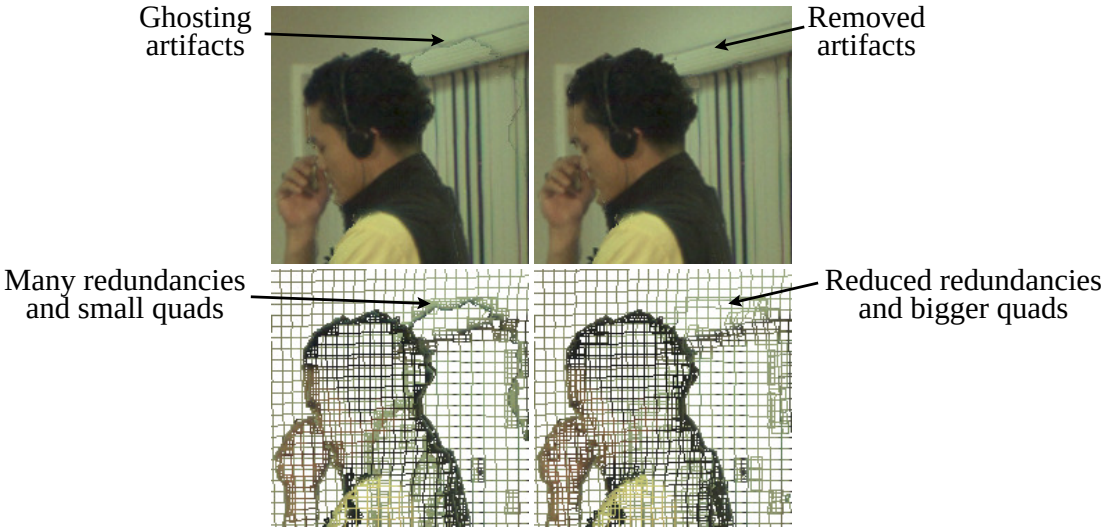
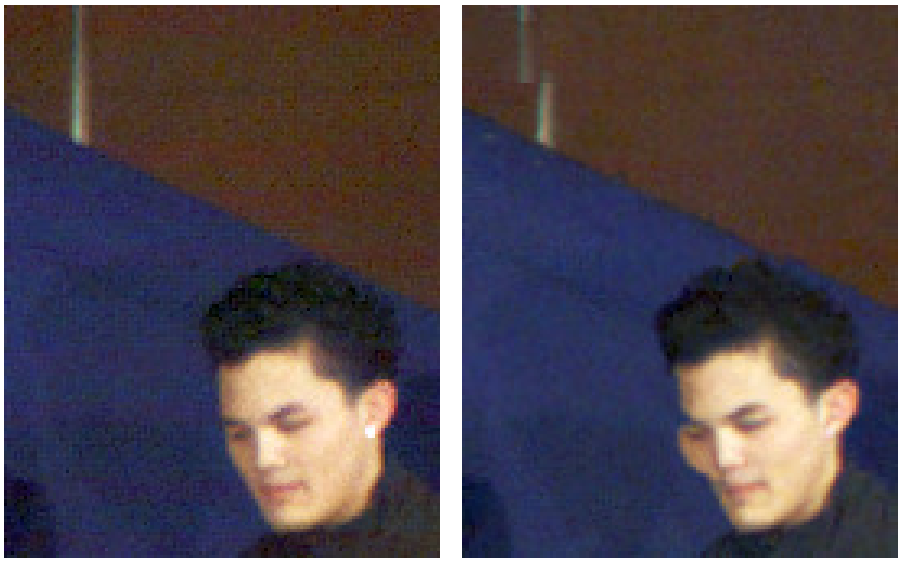


Figure B.6: Réduction des artefacts fantômes. (gauche) Sans la suppression des polygones. (droite) Avec la suppression.

de vues synthétisées donnent une idée de la qualité subjective des images. Figure B.7 montre une vue synthétisée en utilisant la représentation par soupe de polygones. La qualité globale de l'image est bonne et la mesure de PSNR est de 35.17 dB pour cette image. Néanmoins certains désalignements ou certaines déformations apparaissent comme illustré dans la figure B.8.

Figure B.7: Resultat de la synthèse de vue. PSNR = 35.17 dB



(a) *Breakdancers* - vues originale (b) *Breakdancers* - vue synthétisée

Figure B.8: Zoom sur les distorsions observées dans la vue syntétisée.

B.6 Compression de la soupe de polygones

Une nouvelle méthode de compression basée sur la structure en quadtree est présentée. La méthode reçoit en entrée la soupe de polygones, c'est-à-dire plusieurs quadtrees dont certaines branches ont été élaguées pour réduire les redondances. Elle est appliquée sur chaque quadtree indépendamment et pour chaque instant. La prédiction temporelle n'est donc pas utilisée ici. La structure du quadtree est utilisée pour définir de façon récursive les positions et tailles de chaque quad. Ensuite, pour encoder les valeurs de profondeurs des coins des quads, une prédiction utilisant les quads voisins déjà codés est utilisée, et les résidus de prédiction sont quantifiés et codés avec le code ExGolonmb. Enfin, toutes les données sont codées en utilisant le codeur CABAC.

Les performances de cette méthode de compression ont tout d'abord été évaluées avec 3 configurations de la soupe de polygones: avant réduction des redondances; après réduction des redondances; et sans les quads de taille 1 pixel. La dernière configuration est motivée par l'observation que les quads de taille 1 pixel sont très nombreux (30% du nombre de quads) et couvrent une surface très petite. Ainsi, les éliminer doit pouvoir réduire le débit sans trop impacter la qualité. La figure B.9 donne les courbes débit/distorsion correspondant à ces trois configurations. Les résultats correspondent aux attentes, c'est-à-dire que la réduction des redondances permet de diminuer le débit mais a aussi un effet négatif sur la qualité des vues synthétisées, et le fait d'éliminer les quads de taille 1 pixel permet de réduire le débit tout en conservant la qualité des vues synthétisées. En effet, le manque d'information est compensé par les post-traitements comme l'inpainting et le filtrage des contours.

Les performances ont ensuite été comparées à une autre solution actuellement

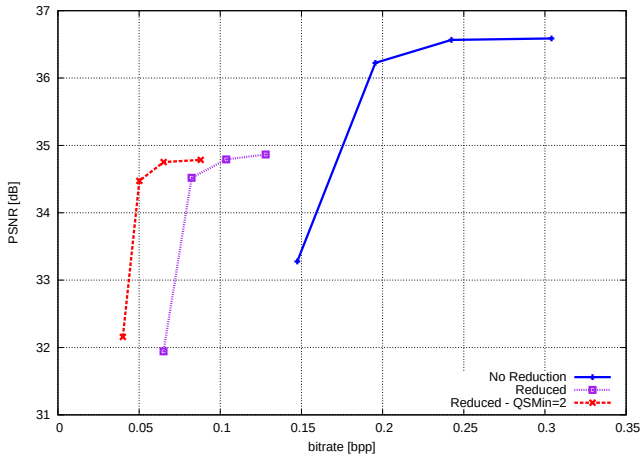


Figure B.9: Trois configurations différentes. Courbes débit/distorsion évaluées sur des vues synthétisées avec la séquence *Breakdancers*.

étudiée par le group MPEG 3DV. Les données utilisées dans cette autre méthode sont MVD. La méthode de compression utilisée pour coder la profondeur est le codage multi-vues H.264/MVC 6.0. Ici la prédiction temporelle est désactivée car elle n'est pas encore prise en compte dans la solution basée sur la soupe de polygones. Enfin la méthode de synthèse de vues est basée point (VSRS 3.0.1: view synthesis reference software). La figure B.10 montre que la solution basée soupe de polygones donne un PSNR légèrement meilleur à moyens et hauts débits. En revanche, cette valeur chute à bas débits.

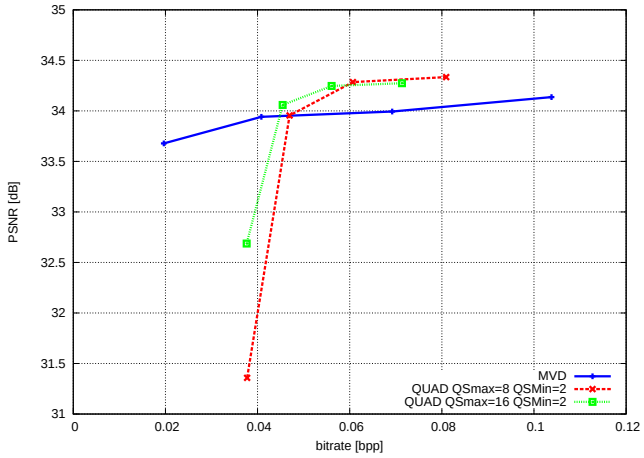


Figure B.10: Courbes débit/distorsion sur vues synthétisées. Comparaison de la méthode basée MVD avec celle basée soupe de polygones. Débit et PSNR moyens calculés sur 25 instants et pour 2 points de vues avec la séquence *Breakdancers*.

B.7 Géométrie déformable

Les artefacts visuels présents dans les vues synthétisées (figure B.8) proviennent principalement des erreurs de modélisations de la géométrie et des caméras. Lorsqu'un seul modèle géométrique est utilisé, les imprécisions ont une influence dans toutes les vues synthétisées. Lorsque plusieurs modèles géométriques sont utilisés, les imprécisions ont une influence moins importante mais les différents modèles peuvent être inconsistants entre eux. A partir de cette constatation, l'idée de la géométrie déformable est de s'adapter à chaque point de vue que l'on veut synthétiser afin de limiter les désalignements de texture et les inconsistances entre modèles.

Cette déformation est guidée par l'alignement de texture après la projection du modèle. Plus précisément, la soupe de polygones est tout d'abord projetée dans chaque vue originale et la correction 3D de la géométrie est calculée en fonction de l'estimation de mouvement 2D entre la vue synthétisée et l'originale (figure B.11). .

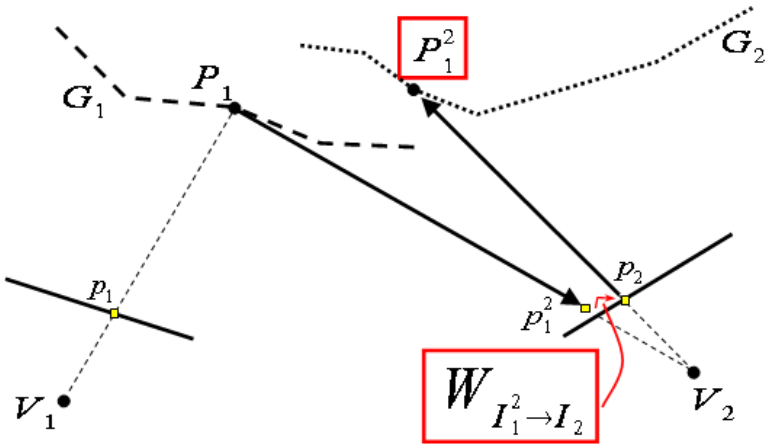


Figure B.11: Calcul de géométrie déformable. P_1 est projeté dans V_2 donnant p_1^2. p_1^2 est ensuite apparié avec p_2 grâce à l'estimation de mouvement. Enfin, p_2 est projeté en 3D en utilisant la géométrie G_2 donnant le point 3D P_1^2.

Une fois la déformation précalculée pour une vue originale, elle est interpolée pour les points de vue virtuels de sorte que les textures projetées dans les vues originales soient toujours alignées et que les transitions d'une vue à l'autre soit douces grâce à l'interpolation (figure B.12).

Pour évaluer les performances de cette méthode, la qualité des vues synthétisées avant et après l'application de la déformation a été évaluée. Les résultats ont montré qu'un gain de 1.53 dB est obtenu lorsque la soupe de polygones est déformée. La figure B.13 montre une partie zoomée d'une vue synthétisée sans la méthode de déformation géométrique (gauche) et avec (droite). On peut y voir que le visage du personnage présentait une déformation pas naturelle et que cela est corrigé grâce à la méthode de déformation de la géométrie. Néanmoins, on peut aussi voir que la ligne blanche qui était coupée au lieu d'être droite n'a pas pu être totalement corrigée.

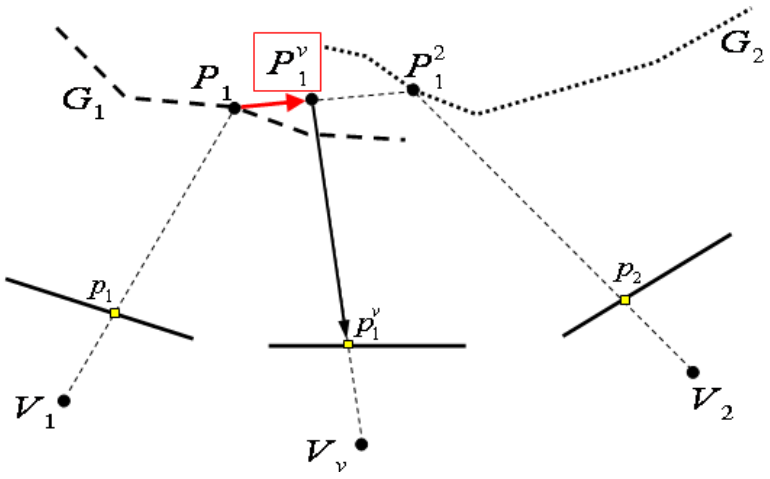
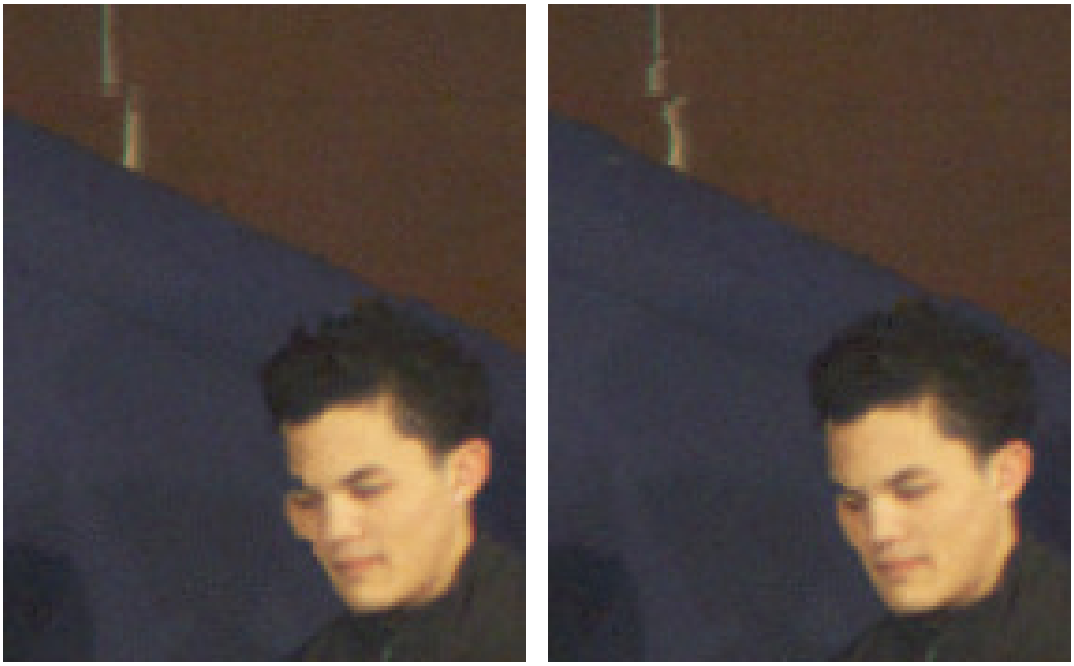


Figure B.12: Synthèse de vues virtuelles avec la géométrie déformable. P_1 est déformé sur une position intermédiaire P_1^v entre P_1 et P_1^2. Ensuite, P_1^v est projeté dans la vue virtuelle donnant le pixel p_1^v.



(a) *Breakdancers* - sans déformation de géométrie (b) *Breakdancers* - avec déformation de géométrie

Figure B.13: Comparaison des vue synthétisées sans et avec la méthode de déformation de géométrie.

B.8 Conclusion

Cette thèse présente une nouvelle représentation pour la vidéo multi-vues. La représentation appelée soupe de polygones est compacte et prend en compte de manière unifiée les problèmes de compression et de synthèse de vue. Elle contient un ensemble de polygones 3D pas nécessairement connectés entre eux et pouvant se superposer. Les polygones sont définis comme des quads 2D avec des valeurs de profondeurs aux quatre coins. Ils sont extraits de cartes de profondeur disponibles en données d'entrée. Les polygones 3D peuvent évoluer dans l'espace et le temps. Enfin, les images originales sont utilisées pour texturer ces polygones.

La construction de la soupe de polygones s'effectue en deux étapes. Tout d'abord, un ensemble de quads est extrait de chaque carte de profondeur par décomposition en quadtree. Ensuite, les redondances inter-vues sont réduites en utilisant un ordre de priorité des quads. Environ 65% de quads sont supprimés grâce à cette étape.

La synthèse de vues virtuelles en utilisant la soupe de polygones utilise des méthodes classiques. Tout d'abord, une redivision des quads est appliquée afin d'éviter l'apparition de fissures entre quads de tailles différentes. Ensuite les polygones sont projetés dans la vue virtuelle et la texture plaquée sur ces polygones en utilisant une carte graphique. Les projections qui se superposent sont combinées par moyenne pondérée. Enfin, les zones vides restantes sont remplies par inpainting et les contours de discontinuités sont adoucis par filtrage. L'évaluation de la qualité des vues synthétisées a montré des vues de bonnes qualité avec un PSNR de 34-35 dB. Cependant certains artefacts apparaissent. Une méthode de déformation géométrique est présentée pour réduire ces artefacts.

Une méthode de compression de la soupe de polygones est présentée. Elle utilise la structure en quadtree pour retrouver efficacement les positions et tailles des quads. Les valeurs de profondeur des coins des quads sont prédites grâce aux quads voisins déjà codés. Les résidus de prédiction sont quantifiés et codés. Toutes les informations sont codées avec CABAC (context adaptive binary arithmetic coding). Les résultats ont montré une légère amélioration de la qualité (+0.3 dB) à moyens et hauts débits.

Enfin, une version dynamique de la soupe de polygone est présentée afin de corriger les désalignements de textures dans les vues synthétisées. Elle consiste à déformer la géométrie 3D en fonction du mouvement 2D calculé entre les vues synthétisées et les vues originales. Les résultats ont montré que les textures sont mieux alignées et que les polygones sont plus consistants entre eux. Une augmentation du PSNR de 1.53 dB est obtenue pour une vue virtuelle de la séquence *Breakdancers*.

Afin d'améliorer cette représentation, nous envisageons d'étudier l'aspect temporel. De la même façon qu'un polygone est déformé en fonction de la vue virtuelle, il pourra aussi se déformer en fonction de l'instant dans la séquence vidéo. Cette nouvelle dimension pourra améliorer la cohérence de la représentation dans le temps et pourra aboutir à une méthode de compression plus efficace.

Bibliography

[AB91] E.H. Adelson and J.R. Bergen. The plenoptic function and the elements of early vision. In *CMVP'91*, pages 3–20, 1991. 29, 48

[AYG+07] A.A. Alatan, Y. Yemez, U. Gudukbay, X. Zabulis, K. Muller, C.E. Erdem, C. Weigel, and A. Smolic. Scene representation technologies for 3DTV - A survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17:1587–1605, Nov. 2007. 25

[Bal05] Raphaèle Balter. *Construction d'un maillage 3D évolutif et scalable pour le codage vidéo.* PhD thesis, Université de Rennes 1, France, may 2005. 125, 129

[BBH08] D. Bradley, T. Boubekeur, and W. Heidrich. Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *CVPR*, 2008. 39, 40, 48

[BBM+01] C. Buehler, M. Bosse, L. McMillan, S.J. Gortler, and M.F. Cohen. Unstructured lumigraph rendering. In *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 425–432, 2001. 19, 98, 108, 125

[BGM06] Raphaele Balter, Patrick Gioia, and Luce Morin. Scalable and efficient coding using 3D modeling. *IEEE Transactions on Multimedia*, 8:1147–1155, dec 2006. 39, 40

[BL09] Harlyn Baker and Zeyu Li. Camera and projector arrays for immersive 3D video. In *IMMERSCOM '09*, pages 1–6, Brussels, 2009. 29, 48

[BVG+07] W.H.A. Bruls, C. Varekamp, R.K. Gunnewiek, B. Barenbrug, and A. Bourge. Enabling introduction of stereoscopic (3D) video: Formats and compression standards. In *ICIP*, pages 89–92, 2007. 33, 35, 48

[Col96] R.T. Collins. A space-sweep approach to true multi-image matching. In *ARPA*, pages 1213–1220, 1996. 38

[CTMS03] J. Carranza, C. Theobalt, M. A. Magnor, and H. Seidel. Free-viewpoint video of human actors. *ACM Trans. Graph.*, 22:569–577, 2003. 38, 39, 48

[Dar09] I. Daribo. *Coding and rendering of 3D video sequences; and applications to Three-Dimensional Television (3DTV) and Free Viewpoint Television (FTV).* PhD thesis, Graduate College of Telecom ParisTech, Paris, France, November 2009. 19, 33, 35

[DBD08] J. Dugelay, A. Baskurt, and M. Daoudi. *3D Object Processing: Compression, Indexing and Watermarking.* Wiley Publishing, 2008. 16, 40

[DDSD03] X. Décoret, F. Durand, F. X. Sillion, and J. Dorsey. Billboard clouds for extreme model simplification. In *SIGGRAPH '03*, pages 689–696, New York, 2003. 43, 44, 48

[DG00] Olivier D. and Pierre-Marie G. Geometric compression for interactive transmission. In *VIS '00*, pages 319–326, Los Alamitos, CA, USA, 2000. 43

[DKM+10] Minh N. Do, Chang-Su Kim, Karsten Müller, Masayuki Tanimoto, and Anthony Vetro. Multi-camera imaging, coding and innovative display: techniques and systems. *Journal of Visual Communication and Image Representation*, 21(5-6):375 – 376, 2010. Special issue on Multi-camera Imaging, Coding and Innovative Display. 9, 160

[Dod05] N.A. Dodgson. Autostereoscopic 3D displays. *Computer*, 38(8):31 – 36, aug. 2005. 20, 22

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20, New York, NY, USA, 1996. ACM. 19, 98, 108, 125

[EDS+08] M. Eisemann, B. De Decker, A. Sellent, M. Magnor, E. de Aguiar, N. Ahmed, P. Bekaert, and H. Seidel. Floating textures. *Computer Graphics Forum (Proc. Eurographics EG'08)*, 27(2), 4 2008. 53, 125, 126, 144

[ESWK04] J. Evers-Senne, J. Woetzel, and R. Koch. Modelling and rendering of complex scenes with a multi-camera rig. In *Conference on Visual Media Production (CVMP)*, 2004. 35

[Fav05] G.E. Favalora. Volumetric 3D displays and application infrastructure. *Computer*, 38(8):37 – 44, aug. 2005. 21

[FKdB+02] C. Fehn, P. Kauff, M. Op de Beeck, F. Ernst, W. IJsselsteijn, M. Pollefeys, L. Van Gool, E. Ofek, and I. Sexton. An evolutionary and optimised approach on 3D-TV. In *In Proceedings of International Broadcast Conference*, pages 357–365, Amsterdam, Netherlands, 2002. 32, 48

[FMZ+08] I. Feldmann, M. Mueller, F. Zilly, R. Tanger, K. Mueller, A. Smolic, P. Kauff, and T. Wiegand. *HHI test material for 3D video.* ISO/IEC JTC1/SC29/WG11MPEG2008/M15413, April 2008. 14

[FTV00] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. A compact algorithm for rectification of stereo pairs. *Mach. Vision Appl.*, 12(1):16–22, 2000. 14

[Fuj94] T. Fuji. *A basic study in the integrated 3-D visual communication.* PhD thesis, University of Tokyo, 1994. 30, 48

[FYT+10] Hisayoshi Furihata, Tomohiro Yendo, Mehrdad Panahpour Tehrani, Toshiaki Fujii, and Masayuki Tanimoto. Novel view synthesis with residual error feedback for ftv. In Andrew J. Woods, Nicolas S. Holliman, and Neil A. Dodgson, editors, *Stereoscopic Displays and Applications XXI*, volume 7524, San Jose, California, USA, 2010. SPIE. 125, 128

[Gal92] Didier J. Le Gall. The mpeg video compression algorithm. *Signal Processing: Image Communication*, 4(2):129 – 140, 1992. 16

[Gal02] F. Galpin. *Représentation 3D de séquences vidéo: Schéma d'extraction automatique d'un flux de modèles 3D, applications à la compression et à la réalité virtuelle.* PhD thesis, Université de Rennes 1, 2002. 39, 40

[GBMD04] F. Galpin, R. Balter, L. Morin, and K. Deguchi. 3D models coding and morphing for efficient video compression. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, CVPR'2004*, pages 334–341, Washington DC, USA, jun 2004. 129

[GBMP04] F. Galpin, R. Balter, L. Morin, and S. Pateux. Efficient and scalable video compression by automatic 3D model building using computer vision. In *Picture Coding Symposium, PCS'2004*, San Francisco, USA, dec 2004. 129

[GFM+07] D. Gallup, J.M. Frahm, P. Mordohai, Q.X. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *CVPR07*, pages 1–8, 2007. 38

[GGSC96] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen. The lumigraph. In *SIGGRAPH '96*, pages 43–54, New York, NY, USA, 1996. ACM. 29, 48

[GKIS05] Stefan Gumhold, Zachi Kami, Martin Isenburg, and Hans-Peter Seidel. Predictive point-cloud compression. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, page 137, New York, NY, USA, 2005. ACM. 43

[GWN+03] Markus Gross, Stephan Würmlin, Martin Naef, Edouard Lamboray, Christian Spagno, Andreas Kunz, Esther Koller-Meier, Tomas Svoboda, Luc Van Gool, Silke Lang, Kai Strehlke, Andrew Vande Moere, and Oliver Staadt. blue-c: a spatially immersive display and 3D video portal for telepresence. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 819–827, New York, NY, USA, 2003. ACM. 43

[JK10] D. Jung and R. Koch. Efficient depth-compensated interpolation for full parallax displays. In *Symposium on 3D Data Processing, Visualization and Transmission, (3DPVT'10)*, Paris, France, May 2010. 21

[JMG09] Vincent Jantet, Luce Morin, and Christine Guillemot. Incremental-ldi for multi-view coding. In *3DTV-Con2009*, apr 2009. 35, 48

[JTC08] ISO/IEC JTC1/SC29/WG11. *Text of ISO/IEC 14496-10:200X/FDAM 1 Multiview Video Coding*. Doc. N9978, Hannover, Germany, July 2008. 31, 112

[JWV+05] N. Joshi, B. Wilburn, V. Vaish, M. Levoy, and M. Horowitz. Automatic color calibration for large camera arrays. Technical Report CS2005-0821, UCSD CSE, May 2005. 14

[KB10] Paul Kerbiriou and Guillaume Boisson. Looking for an adequate quality criterion for depth coding. In *Three-Dimensional Image Processing (3DIP) and Applications*, volume 7526, page 75260A. SPIE, 2010. 35

[KBKL09] A. Kolb, E. Barth, R. Koch, and R. Larsen. Time-of-flight sensors in computer graphics. In M. Pauly and G. Greiner, editors, *Eurographics 2009 - State of the Art Reports*, pages 119–134, CH-1288 Aire-la-Ville, March 2009. Eurographics Association, Eurographics. 15

[KH07] J. Konrad and M. Halle. 3-d displays and signal processing. *Signal Processing Magazine, IEEE*, 24(6):97 –111, nov. 2007. 20

[KLTS06] Sing Bing Kang, Yin Li, Xin Tong, and Heung-Yeung Shum. Image-based rendering. *Found. Trends. Comput. Graph. Vis.*, 2(3):173–258, 2006. 17

[KSM+07] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang. Multiview imaging and 3DTV. special issue overview and introduction. *IEEE Signal Processing Magazine*, 24(6):10–21, November 2007. 9, 160

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987. 38

[LH96] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH '96*, pages 31–42, New York, NY, USA, 1996. ACM Press. 29, 48

[LW85] M. Levoy and T. Whitted. The use of points as a display primitive. Technical report, Computer Science Department, University of North Carolina, 1985. 41, 48

[Mag05] Marcus A. Magnor. *Video-Based Rendering*. AK Peters Ltd, 2005. 9, 17, 25, 160

[Mam08] K. Mamou. *Compression de maillages 3D statiques et dynamiques*. PhD thesis, Université René Descartes - UFR de mathématiques et informatiques, 2008. 40

[MAW+07] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.M. Frahm, R.G. Yang, D. Nister, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *ICCV*, pages 1–8, 2007. 38, 39, 48

[MDMW10] K. Mueller, K. Dix, P. Merkle, and T. Wiegand. Temporal residual data sub-sampling in ldv representation format. In *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1 –4, jun. 2010. 35, 36

[MM09] M. Maitre and M.Do. Shape-adaptive wavelet encoding of depth maps. In *Picture Coding Symposium, Chicago, US*, 2009. 17, 35

[MMS+08] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Muller, P.H.N. de With, and T. Wiegand. The effect of depth compression on multiview rendering quality. In *3DTV Conference*, 2008. 35

[MMS+09] Philipp Merkle, Yannick Morvan, Aljoscha Smolic, Dirk Farin, Karsten Mueller, Peter H. N. de With, and Thomas Wiegand. The effects of multiview depth video compression on multiview rendering. *Signal Processing: Image Communication*, 24(1-2):73–88, 2009. 17

[MMW07] K. Muller, P. Merkle, and T. Wiegand. Compressing time-varying visual content. *Signal Processing Magazine, IEEE*, 24(6):58 –65, nov. 2007. 16

[Mor09] Yannick Morvan. *Acquisition, Compression and Rendering of Depth and Texture for Multi-View Video*. PhD thesis, Eindhoven university of technology, Netherlands, 2009. 17, 18, 35, 91

[MP04] W. Matusik and H. Pfister. 3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. *ACM Trans. Graph.*, 23:814–824, 2004. 28, 29, 48

[MPE06] ISO/IEC JTC1/SC29/WG11 23002-3 MPEG. Auxiliary video data representations. In *Doc. N8038*, Montreux, Switzerland, 2006. 33

[MPE09] ISO/IEC JTC1/SC29/WG11 MPEG. Applications and requirements on 3D video coding. In *Doc. N10857*, London, UK, 2009. 31, 36

[MSD+08] K. Müller, A. Smolic, K. Dix, P. Kauff, and T. Wiegand. Reliability-based generation and view synthesis in layered depth video. In *MMSP*, pages 34–39, 2008. 35, 48, 80

[MSM+04a] K. Mueller, A. Smolic, P. Merkle, B. Kaspar, P. Eisert, and T. Wiegand. 3D reconstruction of natural scenes with view-adaptive multi-texturing. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, pages 116–123, Washington, DC, USA, 2004. IEEE Computer Society. 37, 38, 48

[MSM+04b] K. Muller, A. Smolic, P. Merkle, M. Kautzner, , and T. Wiegand. Coding of 3D meshes and video textures for 3D video objects. In *Proc. PCS 2004, Picture Coding Symposium, San Francisco, USA*, 2004. 48

[MSMW07a] P. Merkle, A. Smolic, K. Muller, and T. Wiegand. Efficient prediction structures for multiview video coding. *CirSysVideo*, 17(11):1461–1473, November 2007. 16

[MSMW07b] P. Merkle, A. Smolic, K. Muller, and T. Wiegand. Multi-view video plus depth representation and coding. *ICIP*, 1:201–204, 2007. 34, 35

[MWS03] D. Marpe, T. Wiegand, and H. Schwarz. Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard. *IEEE Trans. Circuits Syst. Video Techn.*, 13(7):620–636, 2003. 111

[MWTN04] T. Matsuyama, X. Wu, T. Takai, and S. Nobuhara. Real-time 3D shape reconstruction, dynamic 3D mesh deformation, and high fidelity visualization for 3D video. *Comput. Vis. Image Underst.*, 96(3):393–434, 2004. 30, 48

[MZP08] K. Mamou, T. Zaharia, and F. Preteux. Famc: The mpeg-4 standard for animated mesh compression. In *ICIP'08*, pages 2676 –2679, oct. 2008. 40

[MZP09] Khaled Mamou, Titus Zaharia, and Françoise Prêteux. Tfan: A low complexity 3D mesh compression algorithm. *Comput. Animat. Virtual Worlds*, 20(2‐3):343–354, 2009. 40

[NNT07] C. Nitschke, A.i Nakazawa, and H. Takemura. Real-time space carving using graphics hardware. *IEICE - Trans. Inf. Syst.*, E90-D(8):1175–1184, 2007. 30

[OYH09] K. Oh, S. Yea, and Y. Ho. Hole filling method using depth based in-painting for view synthesis in free viewpoint television and 3-d video. In *Picture Coding Symposium, Chicago, US*, 2009. 19, 33, 102

[Paj02] R. Pajarola. Overview of quadtree-based terrain triangulation and visualization. Technical Report (Technical Report UCI-ICS-02-01), Department of Information & Computer Science, University of California, Irvine, 2002. 39, 94, 95

[PK03] Jingliang Peng and C. C. Jay Kuo. Octree-based progressive geometry encoder. In *In Internet Multimedia Management Systems IV. Proceedings of the SPIE*, page 301311, 2003. 43

[PKVG99] M. Pollefeys, R. Koch, and L. Van Gool. A simple and efficient rectification method for general motion. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 496 –501 vol.1, 1999. 14

[PNF+08] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3D reconstruction from video. *Int. J. Comput. Vision*, 78(2-3):143–167, 2008. 39

[PZvBG00] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: surface elements as rendering primitives. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 41, 48

[RL00] Szymon Rusinkiewicz and Marc Levoy. Qsplat: a multiresolution point rendering system for large meshes. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 343–352, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 41, 48

[RNK97] Peter Rander, P J. Narayanan, and Takeo Kanade. Virtualized reality: Constructing time-varying virtual worlds from real events. In *Proceedings of IEEE Visualization '97*, pages 277–283, October 1997. 15

[Say05] Khalid Sayood. *Introduction to Data Compression, Third Edition (Morgan Kaufmann Series in Multimedia Information and Systems)*. Morgan Kaufmann, 3 edition, December 2005. 16

[SCD+06] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 519–528, Washington, DC, USA, 2006. IEEE Computer Society. 18

[SCS05] C. Slinger, C. Cameron, and M. Stanley. Computer-generated holography as a generic display technology. *Computer*, 38(8):46 – 53, aug. 2005. 21

[SGHS98] J. Shade, S. Gortler, L. He, and R. Szeliski. Layered depth images. In *ACM SIGGRAPH*, pages 231–242, 1998. 35, 48

[SJYH10] Feng Shao, Gang-Yi Jiang, Mei Yu, and Yo-Sung Ho. Fast color correction for multi-view video by modeling spatio-temporal variation. *Journal of Visual Communication and Image Representation*, 21(5-6):392 – 403, 2010. Special issue on Multi-camera Imaging, Coding and Innovative Display. 14

[SKS05] O. Schreer, P. Kauff, and T. Sikora. *3D Videocommunication: Algorithms, concepts and real-time systems in human centred communication*. Book, John Wiley & Sons, 2005. 9, 16, 17, 25, 160

[SKS+10] M. Sizintsev, S. Kuthirummal, S. Samarasekera, R. Kumar, H.S. Sawhney, and A. Chaudhry. Gpu accelerated realtime stereo for augmented reality. In *3DPVT'10*, 2010. 30, 31

[SMD+08] A. Smolic, K. Muller, K. Dix, P. Merkle, P. Kauff, and T. Wiegand. Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems. In *ICIP*, pages 2448–2451, 2008. 19, 33, 35, 36, 48, 52, 92, 99, 102

[SMM+09] A. Smolic, K. Müller, P. Merkle, P. Kauff, and T. Wiegand. An overview of available and emerging 3D video formats and depth enhanced stereo as efficient generic solution. In *Picture Coding Symposium*, Chicago, US, 2009. 25, 35, 48, 149

[Smo10] Aljoscha Smolic. 3D video and free viewpoint video - from capture to display. *Pattern Recognition*, In Press, Accepted Manuscript:–, 2010. 25

[SMS+07] A. Smolic, K. Mueller, N. Stefanoski, J. Ostermann, A. Gotchev, G.B. Akar, G. Triantafyllidis, and A. Koz. Coding algorithms for 3DTV - a survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11):1606–1621, Nov. 2007. 40

[Sou10] G. Sourimant. A simple and efficient way to compute depth maps for multi-view videos. In *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2010*, pages 1 –4, jun. 2010. 18, 32

[SS92] R. Sivan and H. Samet. Algorithms for constructing quadtree surface maps. In *the 5th International Symposium on Spatial Data Handling*, volume 1, pages 363–370, Charleston, SC, August 1992. 94

[SSZ01] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *SMBV '01: Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV'01)*, page 131, Washington, DC, USA, 2001. IEEE Computer Society. 18, 32

[Tan06] M. Tanimoto. Overview of free viewpoint television. *Signal Processing: Image Communication*, Volume 21, Issue 6:454–461, 2006. 30, 48

[Tel04] Alexandru Telea. An image inpainting technique based on the fast marching method. *journal of graphics, gpu, and game tools*, 9(1):23–34, 2004. 102

[TFS+08] M. Tanimoto, T. Fujii, K. Suzuki, N. Fukushima, and Y. Mori. *Reference Softwares for Depth Estimation and View Synthesis*. ISO/IEC JTC1/SC29/WG11MPEG2008/M15377, April 2008. 104, 112

[THM10] T. Takai, A. Hilton, and T. Matsuyama. Harmonised texture mapping. In *3DPVT*, Paris, France, May 2010. 128

[TISK10] Mehrdad Panahpour Tehrani, Akio Ishikawa, Shigeyuki Sakazawa, and Atsushi Koike. Iterative colour correction of multicamera systems using corresponding feature points. *Journal of Visual Communication and Image Representation*, 21(5-6):377 – 391, 2010. Special issue on Multicamera Imaging, Coding and Innovative Display. 14

[Tsa86] R.Y. Tsai. An efficient and accurate camera calibration technique for 3D machine vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, Miami Beach, Fl, 1986. 14, 18

[TTN08] Y. Taguchi, K. Takahashi, and T. Naemura. Real-time all-in-focus video-based rendering using a network camera array. In *3DTV-Conference*, 2008. 30, 48

[UCP+09] Matthieu Urvoy, Nathalie Cammas, Stéphane Pateux, Olivier Déforges, Marie Babel, and Muriel Pressigout. Motion tubes for the representation of images sequences. In *Proceedings of ICME'09 IEEE International Conference on Multimedia and Expo*, pages 1–4, Cancun Mexico, 07 2009. 119, 133, 144, 149

[VBK05] Sundar Vedula, Simon Baker, and Takeo Kanade. Image-based spatio-temporal modeling and view interpolation of dynamic events. *ACM Trans. Graph.*, 24(2):240–261, 2005. 129

[WB06] Zhou Wang and Alan C. Bovik. Modern image quality assessment. *Synthesis Lectures on Image, Video, and Multimedia Processing*, 2(1):1–156, 2006. 20

[WDK93] A. Woods, T. Docherty, and R. Koch. Image distortions in stereoscopic video systems. In *Proceedings of SPIE: Stereoscopic Displays and Applications IV*, volume 1915, pages 36–48, 1993. 14

[Whe38] Charles Wheatstone. Contributions to the physiology of vision—part the first. on some remarkable, and hitherto unobserved, phenomena of binocular vision. *Philosophical Transactions of the Royal Society of London*, pages 371–394, 1838. 9

[WLG04] Stephan Würmlin, Edouard Lamboray, and Markus Gross. 3D video fragments: dynamic point samples for real-time free-viewpoint video. *Computers & Graphics*, 28(1):3 – 14, 2004. 43

[WSBL03] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560 –576, jul. 2003. 16

[WWCG07] M. Waschbüsch, S. Würmlin, D. Cotting, and M. Gross. Point-sampled 3D video of real-world scenes. *Image Commun.*, 22(2):203–216, 2007. 15, 41, 42, 43, 44, 48

[WWG07] M. Waschbüsch, S. Würmlin, and M. Gross. 3D video billboard clouds. In *Proceedings of Eurographics*, volume vol. 26, pages pp. 561–569, Prague, Czech Republic, 2007. 44, 45, 48

[YOO06] H. Yamanoue, M. Okui, and F. Okano. Geometrical analysis of puppet-theater and cardboard effects in stereoscopic hdtv images. *Circuits and Systems for Video Technology, IEEE Transactions on*, 16(6):744 – 752, june 2006. 14

[YPYW04] Ruigang Yang, Marc Pollefeys, Hua Yang, and Greg Welch. A unified approach to real-time, multi-resolution, multi-baseline 2D view synthesis and 3D depth estimation using commodity graphics hardware. *International Journal of Image and Graphics (IJIG*, 4:2004, 2004. 14, 30

[YSK+02] S. Yamazaki, R. Sagawa, H. Kawasaki, K. Ikeuchi, and M. Sakauchi. Microfacet billboarding. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 169–180, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association. 45, 46, 47, 48

[YV09] S. Yea and A. Vetro. Multi-layered coding of depth for virtual view synthesis. In *Picture Coding Symposium, Chicago, US*, 2009. 17, 35

[YXDL10] Youwei Yan, Feng Xu, Qionghai Dai, and Xiaodong Liu. A novel method for automatic 2D-to-3D video conversion. In *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2010*, pages 1 –4, jun. 2010. 32

[ZBVH09] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu P. Horaud. Surface feature detection and description with applications to mesh matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Miami Beach, Florida, June 2009. 129

[ZC04] Cha Zhang and Tsuhan Chen. A survey on image-based rendering - representation, sampling and compression. *Signal Processing: Image Communication*, 19(1):1 – 28, 2004. 30

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, 2000. 14, 18

[ZKU+04] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3):600–608, 2004. 15, 32, 33, 35, 36, 48, 99

[ZPvBG02] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 41, 48

[ZXY07] Jinghua Zhang, Jinsheng Xu, and Huiming Yu. Octree-based 3D animation compression with motion vector sharing. In *ITNG '07: Proceedings of the International Conference on Information Technology*, pages 202–207, Washington, DC, USA, 2007. IEEE Computer Society. 43

Abstract

This thesis presents a new representation called *floating polygon soup* for applications like 3DTV and FTV (Free Viewpoint Television). The polygon soup is designed for compactness, compression efficiency, and view synthesis quality. The polygons are stored in 2D, with depth values at each corner. They are not necessarily connected to each other and can be deformed (or *floated*) w.r.t viewpoints and time.
Starting from multi-view video plus depth (MVD), the construction holds in two steps: quadtree decomposition and multi-view redundancy reduction. It results in a compact set of polygons replacing the depth maps while preserving depth discontinuities and geometric details.
Next, compression efficiency and view-synthesis quality are evaluated. Classical methods such as inpainting and post-processing are implemented and adapted to the polygon soup. A new compression method is proposed. It exploits the quadtree structure and uses spatial prediction. Results are compared with an existing MVD compression scheme based on MPEG's H.264/MVC. A slightly higher PSNR value is obtained at medium and high bitrates and ghosting artifacts are greatly reduced.
Finally, the polygon soup is floated according to the desired viewpoint. This view-dependent geometry is guided by motion estimation between synthesized and original views. This method reduces remaining artifacts and improves the final image quality.

Résumé

Cette thèse présente une nouvelle représentation appelée *soupe de polygones déformables* pour les applications telles que 3DTV et FTV (Free Viewpoint TV). La soupe de polygones prend en compte les problèmes de compacité, efficacité de compression, et synthèse de vue. Les polygones sont définis en 2D avec des valeurs de profondeurs à chaque coin. Ils ne sont pas nécessairement connectés entre eux et peuvent se déformer en fonction du point de vue et de l'instant dans la séquence vidéo.
A partir de données multi-vues plus profondeur (MVD), la construction tient en deux étapes: la décomposition en quadtree et la réduction des redondances inter-vues. Un ensemble compact de polygones est obtenu à la place des cartes de profondeur, tout en préservant les discontinuités de profondeurs et les détails géométriques.
Ensuite, l'efficacité de compression et la qualité de synthèse de vue sont évaluées. Des méthodes classiques comme l'*inpainting* et des post-traitements sont implémentées et adaptées à la soupe de polygones. Une nouvelle méthode de compression est proposée. Elle exploite la structure en quadtree et la prédiction spatiale. Les résultats sont comparés à un schéma de compression MVD utilisant le standard MPEG H.264/MVC. Des valeurs de PSNR légèrement supérieures sont obtenues à moyens et hauts débits, et les effets fantômes sont largement réduits.
Enfin, la soupe de polygone est déformée en fonction du point de vue désiré. Cette géométrie dépendante du point de vue est guidée par l'estimation du mouvement entre les vues synthétisées et originales. Cela réduit les artefacts restants et améliore la qualité d'image.